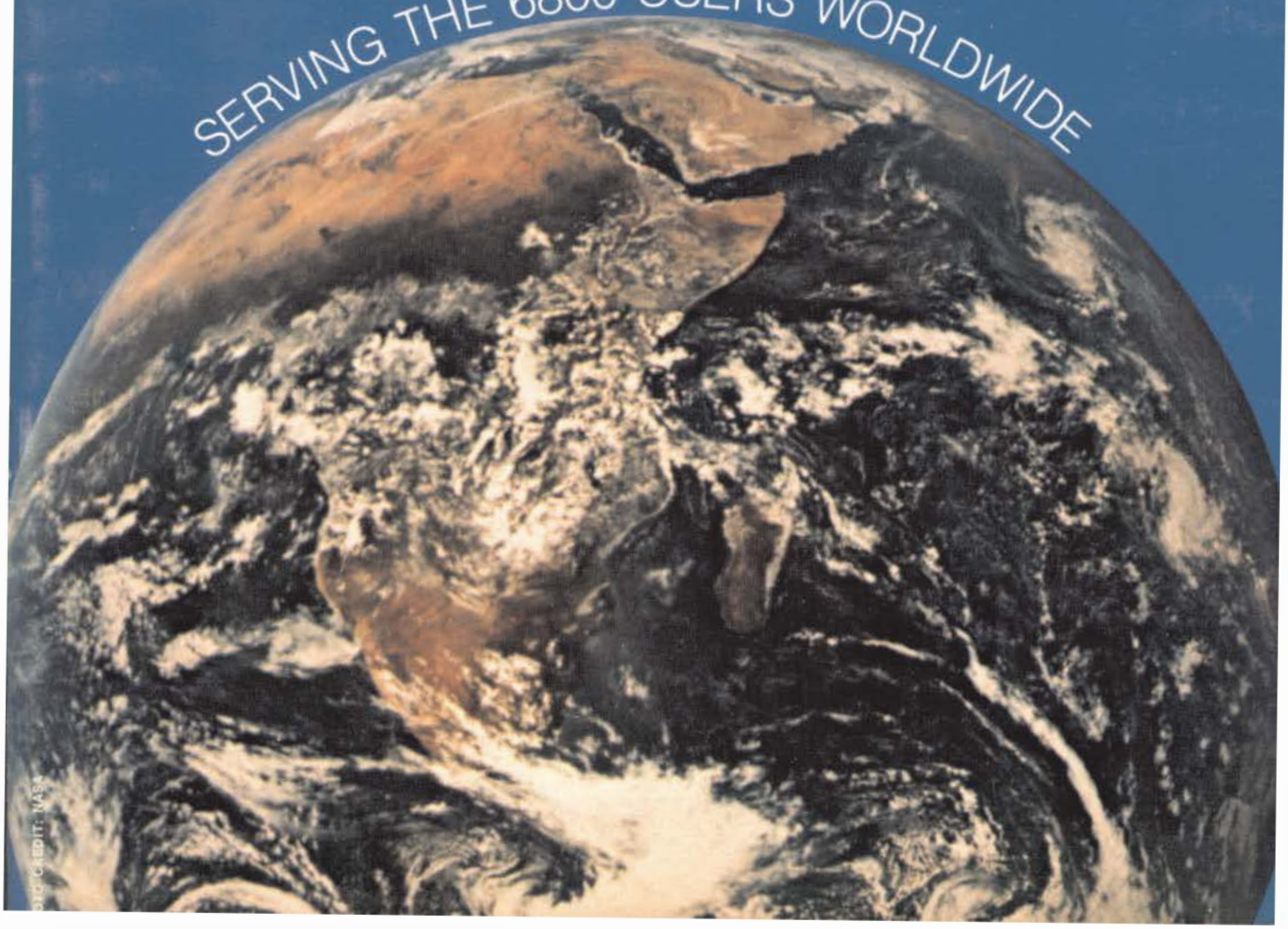
'68' MICRO JOURNAL

$2.50 USA

Australia	A $ 4.00	New Zealand	NZ $ 4.00
Singapore	S $ 8.00	Hong Kong	H $20.00
Malaysia	M $ 8.00		

VOLUME I ISSUE 9 • Devoted to the 68XX User • Nov./Dec. 1979

"Small Computers Doing Big Things."



SYSTEMS - SOLUTIONS

If you have a problem that can be solved by a computer—we have a systems solution.

- Two central processors with maximum RAM capacities of 56K and 384 K bytes
- Three types of disk drives with capacities of 175K, 1.2M and 16M bytes
- Two dot matrix printers with 80 and 132 line capacity
- A Selectric typewriter interface and a daisy wheel printer

Match these to your exact need, add one or more of our intelligent terminals and put together a system from one source with guaranteed compatibility in both software and hardware.

Southwest Technical Products systems give you unmatched power, speed and versatility. They are packaged in custom designed woodgrain finished cabinets. Factory service and support on the entire system and local service is available in many cities.

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Something New on the Horizon from Technical Systems Consultants

Extended BASIC for 6800 and 6809

Finally, a BASIC for serious business applications or scientific programming is available. All the features of our regular BASIC are supported—and more. Floating point calculations are carried out to an internal accuracy of 17 digits. Most math functions are accurate to 16 digits with a minimum accuracy of 13.5 digits. Integer variables have been included to allow fast execution of control loops and array indexing. Even with the double precision math package, this BASIC is still one of the fastest around.

The business programmer will appreciate the versatile PRINT-USING capabilities which include dollar and asterisk fill, trailing minus sign, imbedded commas, and scientific notation. New string functions have been added for string searching (INSTR) and for creating a string which is the date (DATES$). DPEEK and DPOKE are 16-bit peek and poke type functions. The SCALE command has been included to eliminate the round-off errors typically encountered in binary math packages. The INCH$ function allows single-character input from the terminal. Programmer control of control C breaks is also included.

Overall, the Extended BASIC is the most complete BASIC offered for micro users and is only available on FLEX™ disk. A system with at least 32K of user space is recommended. Specify 8" or 5" media (5" 6800 is FLEX™ 2.0) and either the 6800 or 6809 version when ordering.

AP68-12	**6800 Extended BASIC**	**$100**
SP09-6	**6809 Extended BASIC**	**$100**

BASIC Precompiler

This program allows the creation of BASIC programs without the use of line numbers or restrictive two-character variable names. Alphanumeric line and subroutine labels may be used, as well as variable names of any length. Comment lines are marked with non-alphanumerics for easy readability. The output of the precompiler is in the standard BASIC compiled form. This allows applications programs to be written, precompiled, and then distributed in a non-source form. The precompiler can only be used with one of Technical Systems Consultants' BASICs. Specify 8" or 5" (5" 6800 is FLEX™ 2.0) when ordering.

AP68-13	**Single Precision 6800 Precompiler**	**$40**
AP68-14	**Double Precision 6800 Precompiler**	**$50**
SP09-7	**Single Precision 6809 Precompiler**	**$40**
SP09-8	**Double Precision 6809 Precompiler**	**$50**

FLEX is a registered trademark of Technical Systems Consultants, Inc.

technical systems consultants, inc.
Box 2570, West Lafayette, IN 47906
(317) 463-2502

'68' MICRO JOURNAL

Portions of the text of '68' Micro Journal set using the following
6800/2, DMAF1 and CT-82
Southwest Technical Products Corp.
219 W. Rhapsody
San Antonio, TX 78216

Editor, Word Processor and Sort/Merge
Technical Systems Consultants, Inc.
Box 2574
W. LaFayette, IN 47906
"MINIFLEX & FLEX REG."
Technical Systems Consultants, Inc.

Selectric I/O
World Wide Electronics, Inc.
130 Northwestern Blvd.
Nashua, NH 03060

Publisher/Edito
Don Williams Sr.

Executive Editor
Larry E. Williams

Assistant Editor — Software
Mickey E. Ferguson

Assistant Editor — Hardware
Dennis Womack

A ociate Editor — Southwest
Dr. Jack Bryant

Associate Editor — At Large
Dr. Chuck Adams

Associate Editor — Midwest
Howard Berenbon

Contributing Editors
Dr. Jeffrey E. Brownstein
Dale Puckett

Subscriptions and Office Manager
Joyce Williams

Typography and Color Separations
Williams Company, Inc.
Chattanooga, TN 37421

NOV./DEC. Contents

Bryant	8	CRUNCHERS CORNER
Thompson	10	TINY BASIC RENUMBER
Pass	10	BASIC STRING EDITOR
Review	11	SWTPC DMAF2 DISK SYS
	13	CLASSIFIED-MISC
Puckett	14	PASCAL REVIEW
	17	PSYMON (ASSY/SOURCE)
	21-40	NEW PRODUCT
	22	LETTERS
Aldridge	28	INEXPENSIVE DAN
Houng	33	LOGIC GATE TESTER
Ritter	34	PROGRAMMING THE 6809
	37	SOFTWARE CONTEST

Send All Correspondence To:

'68' Micro Journal
3018 Hamill Rd.
PO Box 849
Hixson, Tennessee 37343

— Phone —
Office: 615-870-1993
Plant: 615-892-7544

'68' Micro Journal is published 12 times a year by '68' Micro Journal, 6131 Airways Blvd., Chattanooga, TN 37421. Second Class postage paid at Chattanooga, TN. Postmaster: Send Form 3579 to '68' Micro Journal, PO Box 849, Hixson, TN 37343.

1-Year $14.50 2 Years $26.00 3 Years $36.50

—ITEMS SUBMITTED FOR PUBLICATION—

(Letters to the Editor for Publication) All 'letters to the Editor' should be substantiated by facts. Opinions should be indicated as such. All letters must be signed. We are interested in receiving letters that will benefit or alert our readers. Praise as well as gripes is always good subject matter. Your name may be withheld upon request. If you have had a good experience with a 6800 vendor please put it in a letter. If the experience was bad put that in a letter also. Remember, if you tell us who they are then it is only fair that your name 'not' be withheld. This means that all letters published, of a critical nature, cannot have a name withheld. We will attempt to publish 'verbatim' letters that are composed using 'good taste.' We reserve the right to define (for '68' Micro) what constitutes 'good taste.'

(Articles and items submitted for publication) Please, always include your full name, address, and telephone number. Date and number all sheets. TYPE them if you can, poorly handwritten copy is sometimes the difference between go, no-go. All items should be on 8X11 inch, white paper. Most all art work will be reproduced photographically, this includes all listings, diagrams and other non-text material. All typewritten copy should be done with a NEW RIBBON. All hand drawn art should be black on white paper. Please no hand written code items over 50 bytes. Neatly typed copy will be directly reproduced. Column width should be 3⅛ inches.

(Advertising) Any Classified: Maximum 20 words. All single letters and/or numbers will be considered one (1) word. No Commercial or Business Type Classified advertising. Classified ads will be published in our standard format. Classified ads $7.50 one time run, paid in advance.

Commercial and/or Business advertisers please write or phone for current rate sheet and publication lag time.

SEE GHOST AD PAGES 38 & 43

A.C. POWER CONTROL for ALL COMPUTERS or COMPLETE TURNKEY SYSTEMS

Interface TO the Real World with GIMIX Relay Driver Boards. Connects to any Computer through a 20 ma. current loop (up to 4 Boards - 128 Relays per port).

Interface FROM the Real World with GIMIX

- OPTO BOARDS (up to 34 switch closures with one 8 bit Parallel I/O Port)
- ★ 16 BUTTON KEYPADS
- ★ 35 BUTTON ALPHANUMERIC KEYPADS

A Broad Range of 6800 Systems and Boards Compatible with Other SS50 Products such as SWTP and MSI

MAINFRAME: Includes chassis, power supply, switches, fan and mother board . . **$ 798.19**

16K SYSTEMS: Mainframe, plus 6800 CPU, 16K Static Ram and choice of I/O **$1344.29**
Other packages available.

6800/6809 MOTHER BOARD

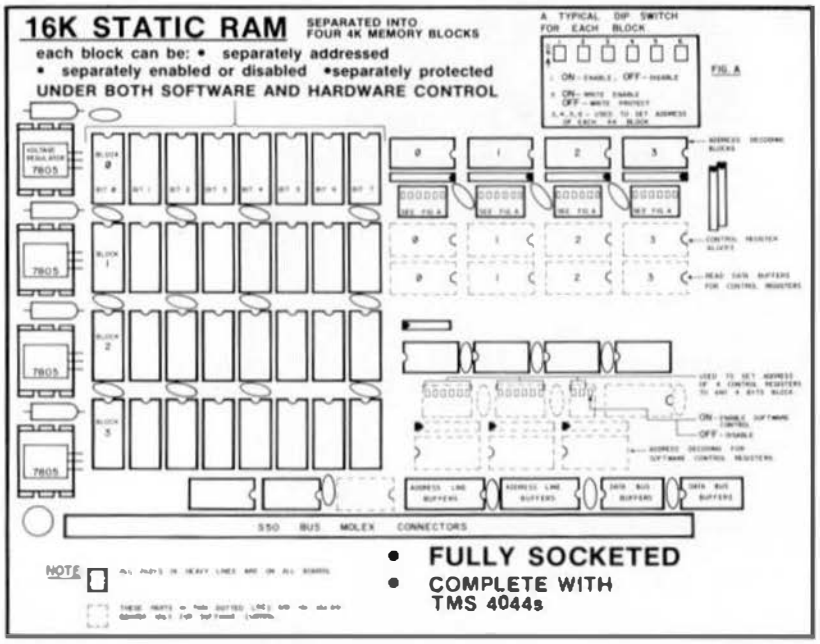
Hardware reconfigurable to give you the utmost versatility for use with various 6800 and 6809, S50 and S50C bus configurations. *Gold plated* pins to insure long lasting electrical contact for protection against corrosion. Fifteen 50 pin slots plus eight DIP-switch addressable 30 pin I/O slots configurable to 4, 8 or 16 decoded addresses. The fully buffered I/O block is addressable by DIP-switch to any 32, 64 or 128 byte boundary and can also be disabled. An on board baud rate generator provides 9 standard baud rates from 75 to 9600. A 14 pin DIP-header allows easy selection of up to five baud rates. The five baud rate lines on the 30 pin bus can be easily disconnected from the 30 pin bus for use as user defined lines (extended address lines etc.) An optional slow I/O circuit, for the 6809 CPU, allows stretching the clock ½ cycle whenever an I/O device is accessed. (This allows, for example, using 1Mh I/O cards with a 2 Mh, 6809 CPU). UD3 and UD2 of the 50 pin bus can be strapped to UD3 and UD4 of the 30 pin bus. A fully shielded, (.090" thick), double sided P.C. board with noise reducing ground lines on the bottom side that separate all data, address, and signal lines, and a full ground plane on the top side. Schmitt trigger buffering of all address, data, and control signals. A 14 position clamping terminal block for all power and other external connections eliminates soldering, crimping or forming of wires.

TI TMS 4044's — 10% SUPPLY
(Not an "equivalent", but the real thing!)

450 ns	$5.90 each	250 ns	$6.90 each
8KPROM BOARD			**$ 98.34**
4K PPD PROM BOARD, Burner and Duplicator			**198.35**
2708's FACTORY PRIME 450 ns		each	**7.90**
64 or 32 x 16 VIDEO BOARD			**198.71**
80 x 24 SUPER VIDEO BOARD with user programmable RAM character generator			**458.76**
Serial I/O's	1 Port **$ 88.41**	4 Port	**198.43**
Parallel I/O's	2 Port **$ 88.42**	8 Port	**198.45**

Add $5. handling charge on orders under $200.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055
Quality Electronic products since 1975.

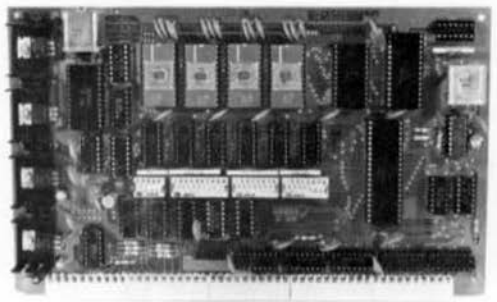


and its also SOFTWARE CONTROLLABLE $368.16

Under software you can program each 4K blocks address and you can enable/disable or write protect each 4K block. The CPU can read the status of the software control registers for each 4K. This allows expansion of memory way beyond 65K; or allows multi-tasking with just one 16K board; all with very little software overhead. *(We have calculated that you can put 13,000 of these boards in a system — that's 208 megabytes, but we do not intend to make either a bus or power supply that size.)* All registers included — no phantom lines needed.

- ★ Facilitates multiprogramming, time-sharing, and sofware development.
- ★ All Gimix memory boards are assembled, burnt-in for 2 weeks, and tested at 2Mh$_z$.

AS ABOVE, BUT WITHOUT SOFTWARE CONTROL AND UNSOCKETED $298.13



GIMIX 6800 CPU BOARD

- ★ 6800 MPU ★ 4K EPROM
- ★ 3 Programmable timers
- ★ 128 byte RAM
- ✓ DIP-switch addressing and enabling or disabling for software use versatility.

This board features:
- ✓ Crystal controlled 6800 MPU using a 6875 clock generator
- ✓ 14411 bit/rate generator with its own crystal that provides baud rates from 110 to 9600.
- ✓ 128 Byte 6810 SCRATCHPAD RAM, which can be DIP-switch addressed to any 128 byte boundary, or disabled.
- ✓ Gold Bus Connectors

Buffered for reliability and data integrity. All address output, data input and output, halt input, clock, reset, control and baud rate output lines are buffered. Manual reset input line is buffered and debounced. IRQ and NMI lines are direct in (6.8K pullup). DMA capability through cycle stealing or halt.

A 4K ROM SECTION THAT HAS:
Sockets for 4 2708 EPROMS which can be DIP-switch addressed to any 4K boundary, or disabled. Dual address switch lets one PROM respond to both E000 and FC00, for MIKBUG compatability. Split address strapping places PROMS at E000, E400, E800, FC00 if desired.

Plus Three Programmable Timers (optional)
Eliminate the need for software timing loops. A 6840 software programmable timer provides 3 independent 16-bit counters which may be used to cause interrupts and/or generate output signals. They can be used individually or in combination. They may be programmed for one-shot timing or for regular intervals. Applications for the timers include frequency measurements, event counting, interval measuring, and similar tasks. They can be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation. DIP-switch addressable to any 8 byte boundary. Additional DIP-switch positions control IRQ or NMI interrupt choices, as well as enabling or disabling the timers.

1-Year $14.50 2 Years $26.00 3 Years $36.50

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card #_____ Exp. Date_____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $_____

Name_____

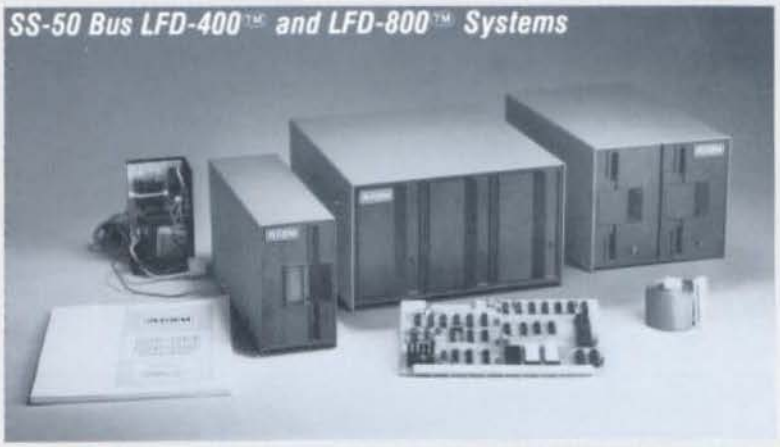
Street_____

City_____ State_____ Zip_____

My Computer Is:_____

68 MICRO JOURNAL
3016 Hamill Road
HIXSON, TN 37343

Welcome to Percom's Wide World



Each LFD mini-disk storage system includes:

- drives with integral power supplies in an enamel-finished enclosure
- a controller/interface with ROM operating system plus extra ROM capacity
- an interconnecting cable
- a comprehensive 80-page users manual

Low-Cost Mini-Disk Storage in the Size You Want.

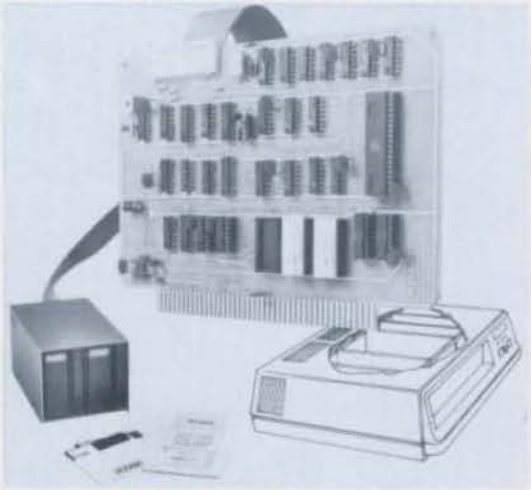
Percom LFD mini-disk drive systems are supplied complete and ready to plug in the moment they arrive. You don't even have to buy extra memory. Moreover, software support ranges from assembly language program development aids to high-speed disk operating systems and business application programs.

The LFD-400™ and -400EX™ systems and the LFD-800™ and -800EX™ systems are available in 1-, 2- and 3-drive configurations. The -400, -400EX drives store 102K bytes of formatted data on 40-track disks, and data may be stored on either surface of a disk. The -800, -800EX drives store 200K bytes of formatted data on 77-track disks.

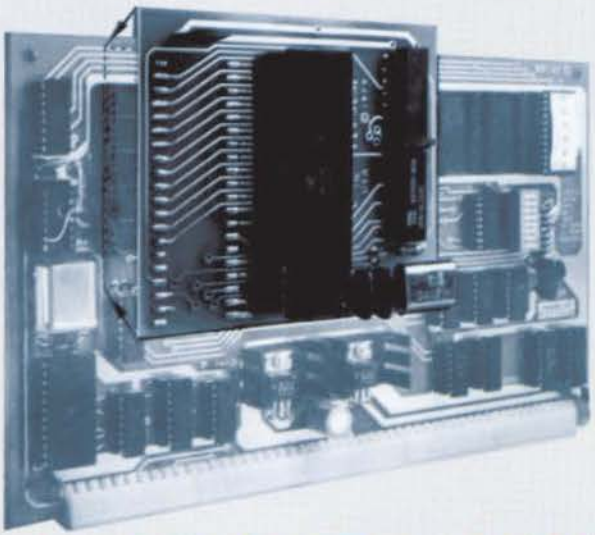
The LFD-1000™ systems (not pictured) have dual-drive units which store 800K bytes on-line. The LFD-1000™ controller accommodates two drive systems so that a user may have as much as 1.6M b tes on-line.

Mini-disk storage system prices:

MODEL	1-DRIVE SYSTEM	2-DRIVE SYSTEM	3-DRIVE SYSTEM
For the SS-50 Bus:			
LFD-400™	$ 599.95	$ 999.95	$1399.95
LFD-800™	895.95	1549.95	2195.95
For the EXORciser* Bus:			
LFD-400EX™	$ 649.95	$1049.95	$1449.95
LFD-800EX™	945.95	1599.95	2245.95
LFD-1000™	(dual) $2495.00	(quad) $4950.00	—



EXORciser Bus LFD-400EX,™ -800EX™ Systems*



Upgrade to 6809 Computing Power. Only $69.95

Although designed with the SWTP 6800 owner in mind, this upgrade adapter may also be used with most other 6800 and 6802 MPUs. The adapter is supplied assembled and tested, and includes the 6809 IC, a crystal, other essential components and user instructions. Restore your original system by merely unplugging the adapter and a wire-jumpered DIP header, and re-inserting the original components. Also available for your upgraded system is PSYMON™ (Percom SYstem MONitor), the operating system for the Percom 6809 single-board computer. PSYMON™ on 2716 ROM costs only $69.95. On diskette (source and object files), only $29.95.

Data Terminal & Two-Cassette Interface — the CIS-30+

- Interface to data terminal and two cassette recorders with a unit only 1/10 the size of SWTP's AC-30
- Select 30, 60 or 120 bytes per second cassette interfacing; 300, 600 or 1200 baud data terminal interfacing
- Optional mod kits make CIS-30+ work with any microcomputer. (For MITS 680b, ask for Tech Memo TM-CIS-30+-09.)
- KC Standard/Bi-Phase-M (double frequency) cassette data encoding. Dependable self-clocking operation
- Ordinary functions may be accomplished with 6800 Mikbug* monitor

Prices: Kit, $79.95. Assembled, $99.95. Prices include a comprehensive instruction manual. Also available: Test Cassette, Remote Control Kit (for program control of recorders), IC Socket Kit, MITS 680b mod documentation and Universal Adapter Kit (converts CIS-30+ for use with any computer).

of 6800 Microcomputing.

6800/6809 SOFTWARE

System Software

6800 Symbolic Assembler — Specify assembly options at time of assembly with this symbolic assembler. Source listing on diskette $29.95

Super BASIC — a 12K extended random access disk BASIC for the 6800 and 6809. Supports 44 commands and 31 functions. Interprets programs written in both SWTP 8K BASIC (versions 2.0, 2.2 & 2.3) and Super BASIC. Features: 9-digit BCD arithmetic, Print Using and LInput commands, and much more. Price $49.95

TOUCHUP™ — Modifies TSC's Text Editor and Text Processor for Percom mini-disk drive operation. Supplied on diskette complete with source listing $17.95

Operating Systems

INDEX™ — This easy-to-use disk-operating and file management system for 6800 microcomputers is fast. I/O devices are serviced by interrupt request. INDEX™ accesses peripherals the same as disk files — new devices may be added without changing the operating system. Other features: unlimited number of DOS commands may be added • over 60 system entry points • display only those files at or above user-specified file activity level • versions available for SWTP MF-68, Smoke's BFD-68 and Motorola's EXORciser* . Price $99.95

MINIDOS-PLUSX™ — An extension of the original MINIDOS™ for LFD-400™ mini-disk systems. MINIDOS-PLUSX™ manipulates files by six-character names. Supports up to 31 files. Resident commands include Initialize, Save, Allocate, Load, Files (directory list), Rename and Delete. Supplied on 2708 ROM with a minidiskette that includes transient utilities such as Copy, Backup, Create, Pack and Print Directory. Price ..... $34.95.

PSYMON™ — Percom SYstem MONitor for the Percom single-board/SS-50-bus-compatible 6809 computer accommodates user's application programs with any mix of peripherals **without** modifying programs. PSYMON™ **also** features character echoing to devices other than the communicating device, sophisticated register and memory dump routines and more. Price (on 2716 ROM) ..... $69.95.

WINDEX™ — Described in detail elsewhere on this page.

Business Programs

General Ledger — For 6800/6809 computers using Percom LFD mini-disk storage systems. Requires little or no knowledge of bookkeeping because the operator is prompted with non-technical questions during data entry. General Ledger updates account balances immediately — in real time, and will print financial statements immediately after journal entries. User selects and assigns own account numbers; tailors financial statements to firm's particular needs. Provides audit trail. Runs under Percom Super BASIC. Requires 24K bytes of RAM. Supplied on minidiskette with a comprehensive users manual. Price ..... $199.95.

FINDER™ — This general purpose data base manager is written in Percom Super BASIC. Works with 6800/6809 computers using Percom LFD-400™ mini-disk drive storage systems. FINDER™ allows user to define and access records using his own terminology — customize file structures to specific needs. Basic commands are New, Change, Delete, Find and Pack. Add up to three user-defined commands. FINDER plus Super BASIC require 24K bytes of RAM. Supplied on minidiskette with a users manual. Price $99.95

Mailing List Processor — Powerful search, sort, create and update capability plus ability to store 700 addresses per minidiskette make this list processor efficient and easy to use. Runs under Percom Super BASIC. Requires 24K bytes of RAM. Supplied on minidiskette with a users manual. Price $99.95.

From the Software Works

Development and debugging programs for 6800 µCs on diskette:

Disassembler/Source Generator $30.95
Relocating Disassembler/Segmented Text Gen $40.95
Disassembler/Trace $25.95
Support Relocator Program $25.95
Relocating Assembler/Linking Loader $55.95
SmithBUG** (2716 EPROM) $70.00

½-Price Special on Hemenway Software!

CP/68‡ disk operating system ..... $ 49.97
STRUBAL+‡ compiler $124.97
EDIT68 text editor ..... $ 19.97
MACRO-Relocating Assembler ..... $ 39.97
Linkage Editor (LNKEDT68) ..... $ 24.97
Cross Reference utility ..... $ 14.97

*Trademark of Percom Data Company, Inc.
• Trademark of Motorola Corporation
‡Trademark of Hemenway Associates Company
**SmithBUG is a trademark of the Software Works Company

And 'looking into' is just what you do with the Electric Window™ as you peer right into memory space where characters are being input and manipulated. Display is memory-resident, programmable and generates up to 24 80-character lines. Other features include:

- standard character generator plus provision for optional special character generator
- dual intensity, high-lighting alphanumeric display
- scrolling by a programmable register • programmable display positioning
- programmable interlaced or non-interlaced scan
- descenders on lower case letters • users manual with application instructions and listing of WINDEX™ driver.

The Electric Window.™ Worth Looking Into. $249.95

WINDEX™ is a fast video display driver program for the Electric Window™. WINDEX™ also features: program and keyboard control of character generators • displayable control characters — under program control • automatic scrolling • a driver routine for the parallel input keyboard feature of the Percom 6809 Single-Board Computer, the SBC/9™ • auto-linking to PSYMON™, the ROM operating system for the SBC/9™ • Prices: ROM version: $39.95; LFD-400™ compatible diskette (source and object files): $29.95.

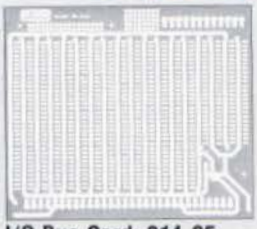
Now Available! the SBC/9™ MPU/Control Computer

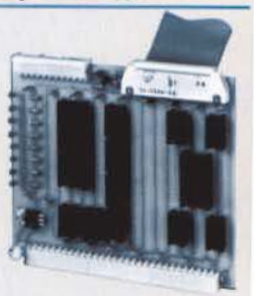
(Single-Board-Computer/6809) — stands alone as a control computer, but also compatible with the SS-50 bus for use as an MPU card. Includes PSYMON™ (Percom SYstem MONitor) in a 1K ROM and provides for additional 1K of ROM. Also includes 1K of RAM. Features: Super Port — provision for multi-address, 8-bit bidirectional data lines • an intelligent data bus for multi-level data bus decoding • an on-board 110-baud to 19.2 kbaud clock generator • extended address capability — to 16 megabytes — without disabling baud clock or adding hardware. And much more. Supplied with PSYMON™ and comprehensive users manual. Price ..... $199.95.

See full page ad elsewhere in this magazine for all of the SBC/9™ features.

Full Feature Prototyping PC Boards

All of the features needed for rapid, straightforward circuit prototyping. Use 14-, 16-, 24- and 40-pin DIP sockets • SS-50 bus card accommodates 34- and 50-pin ribbon connectors on top edge, 10-pin Molex connector on side edge • I/O card accommodates 34-pin ribbon connector and 12-pin Molex on top edge





I/O Bus Card: $14.95

SS-50 Bus Card: $24.95

- I/O card is 1-¼ inches higher than SWTP I/O card • interdigitated power conductors • contacts for power regulators and distributed capacitance bypassing • use wire wrap, wiring pencil or solder wiring • tin-lead plating over 2-oz copper conductors wets quickly, solders easily • FR4-G10 epoxy-glass substrate.

To place an order or request additional literature call toll-free 1-800-527-1592. For technical information call (214) 272-3421. Orders may be paid by check, money order, COD or charged to a VISA or Master Charge account. Texas residents must add 5% sales tax.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

CRUNCHERS CORNER

This monthly column is intended to provide a place for the exchange of ideas on microcomputer arithmetic. A systematic exposition of fixed and floating point arithmetic, hardware and software, algorithms for approximation and so on is planned. Questions and comments submitted to this column can be on any subject relevant to "number crunching," and should be addressed to:

Jack Bryant
Department of Mathematics
Texas A&M University
College Station, Texas 77843

We ask that all correspondents supply their names and addresses.

ALGEBRAIC TO POLISH EXPRESSION TRANSLATION

In the first five issues of '68' Micro Journal, we studied 16 bit two's complement arithmetic. The last installment of this (in July CRUNCHER'S CORNER) featured a complete integer arithmetic package. In addition to the expected arithmetic functions, expression evaluation was provided. The expressions were to be entered (or otherwise transferred to the expression buffer) in reverse Polish notation. This method of expression evaluation is most convenient for hand entry. For example, instead of (A-B)/C=, one enters AB-C/.

In this month's column, we give a simple program to translate from algebraic notation to Polish. Instead of the usual programs in 6800 assembly language which have appeared here, we give the algorithm in two dialects of BASIC. I think this will be of more interest, especially since the translator itself is not all which is required to use these ideas in a high level language. (In particular, the translator does no error checking. That is, it is assumed that the expression is without errors.)*

The Algorithm

The algorithm itself is simple. For the purpose of illustration, it is sufficient to consider the simple case in which the four binary operators +, -, * and / are used, and in which the operands are single alphebetic characters A, B, C,... Right and left parentheses are also operators, of course, but they serve a much different purpose in algebraic expressions. Their main purpose is to alter the implied hierarchy in algebraic notation: * and / before + and -, otherwise left-to-right. The translation requires assigning ranks to each, as follows:

Operator / Operand	Rank
null	-1
A-Z	0
(	1
)	2
+ -	3
* /	4

The null rank indicates the end of evaluation. In the accompanying flowcharts and discussion, the following notation is used:

- L,M -- pointers to the expression
- J -- pointer to the operand stack
- H -- a vector with H(L) the rank of the expression at L
- O -- the operand stack hierarchy

Step 1. Scan the expression left to right and form H. This also furnishes L, one plus the number of operators and operands.

Step 2. Initialize M = 1, J = 2, H(L) = 0 and O(1) = -1.

Step 3. If H(L) = 0, we have an operand: place it on the Polish stack and increment M ; else if H(L) = 2, we have a right parenthesis; decrement J and increment M ; else place the operator on the operator stack, increment J and M and repeat Step 3.

Step 4. Examine the preceeding rank O(J-1) ; if it is less than the current operand hierarchy, place it on the Polish expression stack, decrement J and repeat Step 4; else if M = L , exit to Step 5, else repeat Step 3.

Fancy BASIC Program

The algorithm is coded in a rather nice version** of BASIC in the first listing.

Plain BASIC Program

The next listing is for a much older version# of BASIC (although the string handling may be slightly more convenient for this problem). It requires that strings be dimensioned by the maximum length, does not support if ... then ... else or multiple statements per line; but it does have a nice 'position' function. Shown after the

---

* It is possible that this checking could have been performed by another program. For example, a BASIC compiler could easily be supposed to be accompanied by a BASIC interpreter which detects errors in syntax and even program logic. It is, of course, much easier to write a compiler under the assumption that the source program being compiled contains no syntactic errors.

**TRS-80 (TM) Level II BASIC; I believe it is essentially similar to other recent Microsoft (TM) inspired (or written) dialects of BASIC.

\# This version is the famous (or infamous, if your interest is in structured programming which in this language is essentially impossible) Hewlett-Packard (TM) 9830 BASIC. But I am too critical: the 9830 hardware, particularly the plotter and keyboard layout, are not easily duplicated even six years later.

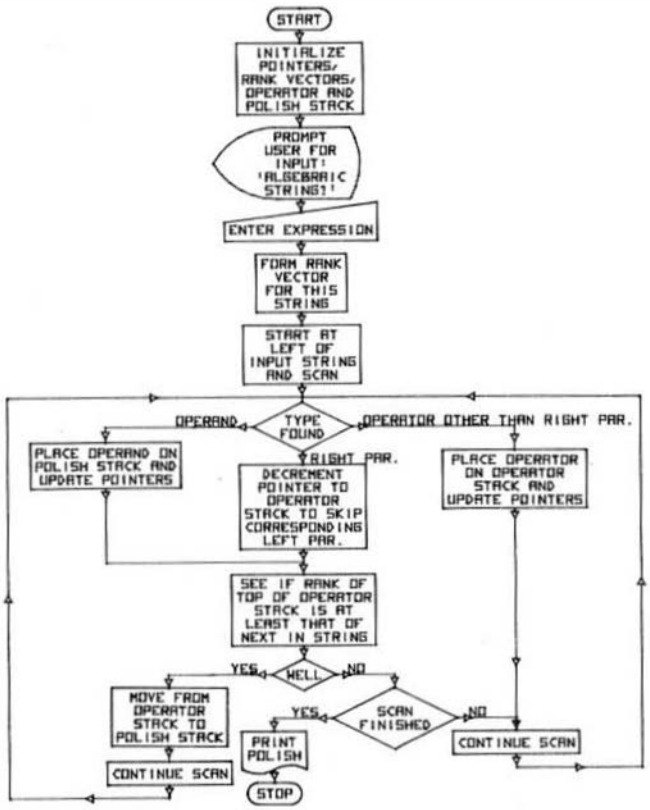
listing are several sample runs of the program.

What's Next?

I suppose the logical thing to do next is to consider parsing, including the error detection problem for expressions. That, of course, has even less to do with number crunching than the topic of this month's column. Although it is an interesting and important diversion, we will not pursue it. Instead, next month we get back on the main track: floating point arithmetic!

A NOTE ABOUT THE FLOW CHARTS

Flow charts seen in this and earlier columns were prepared using an interactive microcomputer flow chart drawing program developed by the author at Texas A&M University with NASA support. The usefullness of flow charts is easy to question, especially when a multi-page set of flow charts is needed to deacribe an algorithm. Thats where the program enters: it puts as much on a page as will fit, takes care of all the centering chores, and so on. With a lot on a page and several levels possible, the main danger of flow charts (that they tend to encourage bottom-up rather than top-down programming) vanishes, and they (the flow charts produced) become a very useful tool. In any case, they furnish a good example of one perhaps unexpected application of microprocessors to a usually tedious drafting and layout planning chore.




```
5 REM CLEAR ROOM FOR STRINGS AND SET UP VECTORS.
10 CLEAR 256: DEFINT A-Z: DIM H(64),O(65),O$(64)
20 REM INITIALIZE VECTORS.
30 FOR I = 1 TO 64: H(I)=0: O(I)=0: NEXT I: P$="": L=0
40 INPUT "ALGEBRAIC STRING";S$
45 REM THIS LOOP (50 & 60) DEFINES H
50 L=L+1: A$=MID$(S$,L,1)
60 IF A$="(" THEN H(L)=1: GOTO 50: ELSE
   IF A$=")" THEN H(L)=2: GOTO 50: ELSE
   IF A$="+" OR A$="-" THEN H(L)=3: GOTO 50 :ELSE
   IF A$="*" OR A$="/" THEN H(L)=4: GOTO 50 :ELSE
   IF A$="" THEN 70 ELSE H(L)=0: GOTO 50
70 H(L)=0: O(1)=-1: M=1: J=2
75 REM STEP 3
80 IF H(L)=0 THEN 90 ELSE
   IF H(L)=2 THEN J=J-1: M=M+1: G TO 80: ELSE
   O$(J)=MID$(S$,J,1): O(J)=H(L): M=M+1: J=J+1: GOTO 80
90 P$=P$+MID$(S$,M,1): M=M+1
95 REM NOW STEP 4.
100 IF O(J-1)>=H(M) THEN P$=P$+O$(J-1): J=J-1: GOTO 100: ELSE
    IF M>=L THEN PRINT "POLISH STRING-->  ";P$: GOTO 10: ELSE 80
```

```
10 DIM S$(64),O$(64),P$(64),H(65),O(64),F$(6),A$(1)
20 FOR I=1 TO 64
30 H(I)=O(I)=0
40 O$(I)=" "
50 P$(I)=" "
60 NEXT I
70 F$="()+-*/"
80 L=0
90 DISP "ALGEBRAIC STRING"
100 INPUT S$
110 DISP
120 PRINT "ALGEBRAIC STRING:  ";S$
130 L1=LEN(S$)
140 L=L+1
150 IF L>L1 THEN 280
160 A$=S$(L,L)
170 GOTO POS(F$,A$(1),1) OF 200,220,240,240,260,260
180 H(L)=0
190 GOTO 140
200 H(L)=1
210 GOTO 140
220 H(L)=2
230 GOTO 140
240 H(L)=3
250 GOTO 140
260 H(L)=4
270 GOTO 140
280 H(L)=0
290 O(1)=-1
300 M=K=1
310 J=2
320 IF H(M)=0 THEN 420
330 IF H(M)=2 THEN 390
340 O(J)=H(M)
350 O$(J,J)=S$(M,M)
360 J=J+1
370 M=M+1
380 GOTO 320
390 J=J-1
400 M=M+1
410 GOTO 450
420 P$(K,K)=S$(M,M)
430 M=M+1
440 K=K+1
450 IF O(J-1)<=H(M) THEN 480
46  IF M>=L THEN 520
470 GOTO 320
480 P$(K,K)=O$(J-1,J-1)
490 K=K+1
500 J=J-1
510  OTO 450
520 P$(K+1)=""
530 PRINT "POLISH STRING -->  ";P$
540 PRINT
550 GOTO 20

ALGEBRAIC STRING:  A+S
POLISH STRING -->  AS+

ALGEBRAIC STRING:  A-S*(Z+X)
POLISH STRING -->  ASZX+*-

ALGEBRAIC STRING:  ((((((((((A))))))))))
POLISH STRING -->  A

ALGEBRAIC STRING:  A+B*(S+D*(F+G*(H+J*K)))
POLISH STRIN  -->  ABSDFGHJK********

ALGEBRAIC STRING:  (((((A-B)-C)-D)-E)-F)-G
POLISH STRING -->  AB-C-D-E-F-G-

ALGEBRAIC STRING:  A-B-C-D-E-F-G
POLISH STRING -->  AB-C-D-E-F-G-
```

BASIC STRING EDITOR

Dr. E. M. Pass
1454 Latta Ln.
Conyers, GA 30207

Recently I had the necessity of converting a system composed of several lengthy BASIC programs to do a similar job. The primary changes involved converting file names and literal strings from those used in the old system to those used in the new. Since the BASIC that I use (MSI 12K DISK BASIC) embodies a line-oriented editor and has a non-transparent disk format for programs, the problem was non-trivial. After several attempts, I developed the program listed below.

The program which I call SCANNER resides in the last portion of MIKBUG scratchpad memory, which is one of the few areas of memory which BASIC does not use. To use it, one should first load BASIC, then the BASIC program to be changed. To escape from BASIC, the user may type PATCH. SCANNER should not be loaded directly in place from disk but may be loaded from tape or in a different area of memory and then relocated. This is because some DOS functions use the same portion of scratchpad for stack and temporary storage and this would destroy SCANNER. Then, using MIKBUG or SWTBUG, the user may enter the following data:

```
A002-3      starting address for search (0100)
A004-5      ending address for search (7F00)
A006        string length
A016...     original string
A026...     new string
```

If using SWTBUG, all that is necessary then is to type

```
J A04A
```

to execute the program, wait for the $ to be printed, and type

```
G
or
J 0103
```

to return to BASIC. If using MIKBUG, it will be necessary to type

```
M A048
```

and place A04A in memory locations A048 and A049, type

```
G
```

to execute the program, wait for the * to be printed, type

```
M A048
```

again and place 0103 in memory locations A048 and A049, then type

```
G
```

to return to BASIC.

At this point, the string will be changed. If other strings were required to be changed, they may be changed before returning to BASIC by repeating the changes to A006, A016 - A025, and A026 - A035 and executing the program for each case. The boundaries for the search are somewhat generous at 0100 to 7F00, but will work. The primary danger when using this program is in incorrectly matching and modifying strings in the BASIC program or in the BASIC interpreter itself. Better search boundaries would solve the second problem, but only intelligent searching will solve the first problem.

This routine may also be used to modify arbitrary strings in memory regardless of whether BASIC is loaded or not. In this case the search boundaries should be set as follows:

```
A002 - 3     starting address
A004 - 5     ending address - string length + 1
```

A listing of the program and a sample execution follow.

```
M A002
*A002 01
*A003 00
*A004 01
*ww05 FF
*A006 04
*
*M A016
*A016 13
*A017 14
*A018 15
*A019 16
*A01A 17
*
*M A026
*A026 30
*A027 31
*A028 32
*A029 33
*A02A 34
*A02B 35
*
*M 0100
*0100 13
*0101 14
*0102 15
*0103 16
*0104 17
*0105 18
*0106 13
*0107 14
*
*M A048
*A048 A0
*A049 4A
*
*G
*
*M 0100
*0100 30
*0101 31
*0102 32
*0103 33
*0104 17
*0105 18
*0106 13
*0107 14
```

```
0 0010                    MAN    SCANNER
0 0015                    OPT    O
0 0020 A04A               ORG    $A04A
0 00 0 A04A FE A002 START LDX    $A002
0 0040 A04D 20 04         BRA    OVER
0 0050 A04F FE A000 BACK  LDX    $A000
0 0060 A052 08            INX
0 0070 A053 8C A004 OVER  CPX    $A004
0 0080 A056 26 03         BNE    CONT
0 0090 A058 7E E0D0       JMP    $E0D0
0 0100 A05B FF A000 CONT  STX    $A000
0 0110 A05E F6 A006       LDA B  $A006
0 0120 A061 8E A015       LDS    #$A015
0 0130 A064 32      FIND  PUL A
0 0140 A065 A1 00         CMP A  X
0 0150 A067 26 E6         BNE    BACK
0 0160 A069 08            INX
0 0170 A06A 5A            DEC B
0 0180 A06B 26 F7         BNE    FIND
0 0190 A06D FE A000       LDX    $A000
0 0200 A070 F6 A006       LDA B  $A006
0 0210 A073 8E A025       LDS    #$A025
0 0220 A076 32      FIX   PUL A
0 0230 A0 7 A7 00         STA A  X
0 0240 A079 08            INX
0 0250 A07A 5A            DEC B
0 0260 A07B 26 F9         BNE    FIX
0 0270 A07U 7E A04F       JMP    BACK
0 0280                *
0 0290                *   A002-3 = START ADDRESS
0 0300                *   A004-5 = END ADDRESS - LENGTH + 1
0 0310                *   A006   = LENGTH OF STRING
0 0320                *   A016.. = ORIGINAL STRING
0 0330                *   A026.. = NEW STRING
0 0340                *
0 0350                        END

TOTAL ERRORS 00000
```

TINY BASIC RENUMBER

N. J. Thompson
1615 Wilder Ln.
Honolulu, Hawaii 96822

*** * RENUMBER YOUR * ***
*** * TINY BASIC TEXT EDITOR * ***

A TINY-BASIC, SUCH AS TOM PITMAN'S, HAS CAPABILITIES AS A TEXT EDITOR, FOR TWO GOOD REASONS.

FIRST, IT DOES NOT CHECK SYNTAX UNTIL A LINE IS EXECUTED. THIS MEANS THAT YOU CAN WRITE SOMEONE A LETTER IN PLAIN LANGUAGE INTO TINY BASIC, AND BASIC WILL ACCEPT IT.

SECOND, BASIC OF COURSE PERMITS THE MAIN FEATURES OF TEXT EDITING. BY PERMITTING REPLACEMENT OF LINES, AND INSERTING OF LINES BETWEEN PREVIOUSLY NUMBERED LINES.

THERE ARE TWO REMAINING STEPS THAT WOULD BE NICE FOR USING TINY AS A SMART LETTER-PROCESSOR. THE FIRST IS A COMMAND TO LIST WITHOUT LINE-NUMBERS. I HAVE WRITTEN MACHINE-LANGUAGE PROGRAMS, AND EVEN TINY-BASIC PROGRAMS, TO DO THIS. BUT THAT IS NOT THE SUBJECT OF THIS ARTICLE.

THE SECOND ENHANCEMENT, WHICH IS THE SUBJECT OF THE FOLLOWING, IS RENUMBER. IT WOULD BE NICE TO TEACH THE MACHINE TO RENUMBER YOUR TEXT IN TINY-BASIC, TO PERMIT INSERTING MORE TEXT IN THE MIDDLE.

RENUMBERING A BASIC PROGRAM INVOLVES TWO STEPS. THE FIRST IS CHANGING ALL THE LINE NUMBERS, AND THE SECOND IS LOOKING UP ALL THE GOTO AND GOSUB INSTRUCTIONS, BOTH BEFORE AND AFTER THE LINE NUMBERS ARE CHANGED, AND CHANGING THE PARAMETERS OF THE THE GOTO STATEMENTS TO POINT TO THE NEW LINE NUMBERS.

BUT IF WE ARE JUST WRITING SOMEONE A LETTER, AND DONT PLAN TO EXECUTE THE CODE, THE SECOND REQUREIMENT DOESN'T EXIST. YOU CAN SCREAM THRU, CHANGING THE LINE NUMBERS TO ANY PATTERN, WITHOUT ANY DIRE CONSEQUENCES.

HERE'S HOW TO RENUMBER THE LINES:

```
CE 11     LDX #11 : FIRST NEW LINE #
FF 00B2   STX 00B2 : IN TINY VARIABLE Y
CE #00B8  LDX #00B8 : START OF TINY PRGM.
FF 00B4   STX 00B4 : SAVE IN TINY Z

      : LOOPBACK POINT
FE 00B4   LDX 00B4 : LINE NUMBER ADDX
EE 00     LDX 00,X : LINE NUMBER



26 01     BNE AHEAD : LINE # = 0 ?
39        RTS  : YES, DONE. RETURN

FE 00B4   LDX : ADDX AGAIN
96 B2     LDAA : HIBITS OF NEW LINE #
A7 00     STAA 0,X : INTO PLACE.
96 B3     LDAA : LOBITS ALSO
A7 01     STAA 01,X
08        INX  : PASS OVER LINE NUMBER

        : GO ON TO NEXT TINY LINE
08        INX  : PASS OVER CHARACTER
A6 00     LDAA 00,X : IS THIS CHARACTER
81 0D     CMPA #0D : - A RETURN?
26 F9     BNE : NO, KEEP CHUGGING THRU
08        INX : YES. PASS OVER CR.
FF 00B4   STX 00B4 : POINTER TO NEXT LINE

4F        CLR A : TO INCREMENT LINE #
C6 08     LDAB #0A : BY TEN.
DB B3     ADDB B3 : ADD LOWBITS.
99 B2     ADCA B2 : AND HIDITS.
97 B2     STAA B2 : STORE NEXT LINE
D7 B3     STAB B3 : NUMBER IN TINY Y.
20 D4     BRA : TO LOOPBACK POINT.
```

THE ABOVE RELOCATABLE CODE CAN GO ANYWHERE THAT WON'T DISTURB TINY. THERE IS PROBABLY SPACE DOWN A WAYS FROM THE TOP OF MEMORY THAT TINY USES.

THE CODE ASSUMES, IN THE THIRD LINE, THAT YOUR BASIC PROGRAM STARTS AT 00B8. MAYBE YOURS STARTS SOMEWHERE ELSE.

THE FIRST LINE SETS THE INITIAL VALUE FOR RENUMBERED LINES AT 17 (HEX 11). I RENUMBER FROM 17 BY TENS TO REMIND ME AT A GLANCE WHICH NUMBERS ARE RENUMBERED AS I CONTINUE TO EDIT.

YOU CAN CALL THIS SUBROUTINE FROM YOUR MONITOR, OR FROM TINY WITH THE IMMEDIATE CALL X=USR(NNNN), WHERE NNNN IS THE DECIMAL LOCATION OF THIS CODE.

YOU WILL FIND THE BASIC LINE NUMBERS INSTANTLY RENUMBERED.

AND EVEN IF THE BASIC WAS A PROGRAM, WITH GOTO AND GOSUB PROBLEMS, ITS EASIER TO SUBSEQUENTLY RENUMBER THE JUMPS THAN TO RETYPE THE WHOLE PROGRAM WHEN YOU NEED TO INSERT MORE STATEMENTS.

HELP

Has anyone interfaced the SWTP MF-68 disk to the Motorola MEK6800 D-2 using the MEK 6800AB motherboard? I'd like to know of any problems and about system compatibility.
Wendell Anderson, 2210 Blue Berry Hill, San Antonio, TX 78232

SWTPC DMAF2 DISK SYSTEM

REVIEW

As many of you realize, 68 Micro Journal™ has grown rapidly since our first issue, of February this year. We es e computer magezine, unlike most of the other computer magazines, use computers, 6800 end 6809, to do most ell our text editing, composition end copy output. All accomplished with standard 6800 end 6809 computers, just like those you our readers own end use. We even have our major accounting date processing done by 6800 end 6809 computers. In addition to the above we maintain e complete date file, of ell our subscribers, advertisers end authors, on 6800 end 6809 computers, egain just like yours. In all we have nine 6800 end/or 6809 computers in our system, working computers that is, all major makes (except two, SSB and MSI), these we hope to add to our staff soon. Right; 'steff' each one has e name, given by various human staff members. They ell work, there ere no 'bugs' or 'funny happenings' in any of our TTL staff. I read the laments of other computer magazine staff writers end publishers, using mostly S100 bus machines(?) or e gaggle of those expensive toys of different gender, misnomered es computers, then I look around et ell the 6800 end 6809 machines we use daily end get e pretty satisfied feeling.

All this may seem inappropriate to the theme of this review, except for the fact that the SWTPC DMAF2 Floppy Disk System helps us do our work even better. At least it makes it e lot easier.

The only problem we have experienced with using the same computers that are advertised in 68 Micro Journal™, was the need for more mass storage. Especially our inhouse subscriber list (our mailers have the subscriber list also, et least each has a part of it). When one of you call concerning your subscription, almost always we can have one of our computers scan for your file end get the problem settled, while you ere still on the phone. Try that with one of the other magezines! I did e couple years back, seems they kept charging my Master Charge every month for e 3 year subscription. After it zoomed past the $100+ mark I started calling; my letters went unanswered. Not once in 4 calls could they get it straight, the story was always the same, will 'try' to get it out of the computer end let you know. Finelly after e person-to-person call to the lady assistent of the publisher, did I get it streight. Even then they were not sure. I was asked, "well how much have we overcharged you"? I was even e slight bit hazy by this time (6 weeks of calling), I replied, "oh, something over a hundred dollars, around e hundred and twenty-seven I believe". I received e check for exactly $127.00 e week or so later. I got out all my back Master Charge billings end ran en accurate total, not including all the long distance calls, it came to $132.50. I don't think they ever did get it straight on their end, computer always down, or something like that. Many times es we use our computers to straighten out misunderstandings (yes, we have them also, only difference is we get streight answers from our computers) I think back on that episode and feel sorry for all those trying to do serious work with toys rather then tools.

The SWTPC DMAF2 Floppy Disk System makes any 6800/09 computer system e tool, in the exact sense of the word. As I have said, all ours worked, we just needed more mass storage. For instance; We use a 6809 machine (SWTPC) to

maintain all the data files described above. Up until a few weeks back we were restricted to 8" double sided floppy disk, single density. This prevented us from maintaining all these files on one disk (with two backups current at all times). When we desired to run down a name or address or some other item on disk, we had to open and close various data files and change disk, each time. That got to be a pain. We needed more contiguous mass storage.

Since converting over to the new SWTPC DMAF2 dual density system, we have each file category on one disk. In fact it gives us approximately one and a quarter million bytes of actual data storage, per disk. So counting both drives we have approximately 2.5 megabytes on line at all times.

The SWTPC DMAF2 dual density disk system uses the Qume 8" double sided disk drives. It may be operated in either single or double density mode. It will transfer data from a single density to a double density disk or vice-versa. The controller looks at the disk formatting and determines if it is double or single density. Based on this determination it configures itself to read or write in the proper mode. It is all automatic and requires no operator intervention. The only time the operator must tell it about single or double density is when a FLEX™ 'NEWDISK' operation is done. It formats each disk either single or double, depending on the needs of the operator. Some disk (backups) we 'NEWDISK' are single density, this way our 6800 systems can use them also, for text type files. However, there is one kink when using text files from one system to the other.

The older SWTPC DMAF1 system uses the WD 1771 disk LSI controller IC. The DMAF2 system uses the WD 1791 LSI controller IC. The WD 1791 cannot read disk created using the 1771. The 1771 can read both format types (the 1771 writes 00's in some gap areas, the 1791 writes 11's in these areas), the 1791 can read only disks formatted with 11's. This poses a problem for the user who has only the one disk system. It means that he must have a disk, already 'newdisked' by the 1791 to get his system up an running. He (or she) can then backup (6800) or copy (no backup command with 6809 FLEX™) to the new 1791 formatted disk, a systems disk or any other type disk desired. All recent 6800 software from SWTPC and TSC is formatted in the 11 format, so both controllers can use the disk. Older disk from TSC (those we have back beyond April 79) are of the 00 format and cannot be copied on the DMAF2 system.

Those upgrading from the DMAF1 to the DMAF2 might have a problem unless they request a formatted disk from the vendor, when purchasing a seperate DMAF2 dual density controller board ($395.00). We could not bring our system up on the 6800 because all our disks were formatted with the 1771, the 1791 was having no part of that. We had an easy out; we merely newdisked a disk on the 6809 machine (running a DMAF2) then copied our system (6800) disk onto the new disk while still running on the 6809 machine. Once we had a new system disk for the 6800, with gap recordings of 11, we then went back to the 6800 DMAF2 system and went from there. The 1771 was happy with the 11's. Of course if you are bringing up the system as a whole, there should be no problem as the disk you receive with the complete system has the 11 gap format.

The DMAF2 double density disk system should work on any S50 bus computer. It (controller board) comes equipped so that it can be jumper programmed for the 2 mhz machines also. There are five different options, depending on machine speed and type of memory installed. 6800 users should have 8K of memory located from $A000 thru $BFFF, for the disk operating system. Also a minimum of 12K of contiguous memory should be available in the 6800 machines, from 0-12K and 20K in a 6809 machine.

Two diagnostics are given, REGTEST verifies that the controller registers are rsponding properly. STEPTEST verifies the track selection circuitry of the controller and the response of the disk drives.

The SWTPC DMAF2 system is composed of four major units. The disk controller board interfaces the computer to the disk drives. No component layout drawings, schematics or parts list are supplied. The board is quite complex and requires some special equipment as well as a good dual trace scope to align or troubleshoot. Field service is not recommended by SWTPC.

The FO-M Motor Control board turns the disk off and on, by supplying a/c power to the drives, as determined by a timer on the controller board. This power is optically coupled by two triacs. This isolates the system ground from the a/c power lines.

The P-20 Power Supply is a simple but adequate 7 and 32 VAC source for the drives. Full wave rectifiers pass +5V dc and +24V dc to the drive logic boards. Power for the controller board is supplied by the computer and onboard regulated.

Pre-compensation is accomplished by the controller board logic when double density mode is used. Jumper selection is available on the controller board to compensate for slow memory in the computer. This is for those 6809 machines with slow memory enable lines. Jumpers are available for I/O addressing for either the 6800 (32K) or the 6809 (56K). There are other options available on the board that are for future hardware upgrades.

The instructions contain information on how to modify various memory boards for above 32K addressing. As this machine comes factory built there are not some of the more detailed instructions, as is found with most kits. The standard configuration is two drives, however, the system can handle up to four drives, with existing software.

We have been using our SWTPC dual density, dual disk system for over a month now and have experienced no problems whatsoever. It is fast (appx. 4 times faster) than our 5" systems and now allows us to handle our long data files in a more efficient manner.

Overall the SWTPC DMAF2 system appears ideal for small (and some large) business or industrial applications. It is of rugged construction and uses quality components throughout. The entire system is exhaust fan ducted and runs very cool. The fan makes very little noise and creates no distraction in our office.

Additional information can be secured by contacting:

Southwest Technical Corporation
219 W. Rhapsody
San Antonio, Texas 78216

A 68 Micro Journal™ Lab rating of AAA.

Rating Scale:
AAA - Excellent
AA - Good
A - Fair (could be better but works)
P - Poor (may not always work properly)
X - Not recommended for children
(or anything else!)

SMALL BUSINESS ADVERTISING

It has been noted that there are many small 6800 businesses that can not substain a prolonged advertising campaign. In order to assist these in establishing 'name recognition' and at the same time let our readers know what they have to offer, we are starting, with the February '80 issue, a 'BUSINESS CARD' advertising page.

The plan works something like this; First we will place your ad, which will be a direct copy of your business card, on a special page with other 'BUSINESS CARD' advertisers. The ad must be exactly as you submit your business card. We will not allow any changes unless you change your business card, then the new card copy will replace the existing card advertising. There will be a one time charge of $7.50 to "CHANGE" card copy, as indicated above.

It will be required that you pre-pay for 3 months of advertising, at the rate of $39.95 per month, for a total of $119.85. Payment in advance is required.

As is our existing policy to protect our readers, the following requirement applies to this type advertising, as all other advertising:
All advertisers taking advantage of this offer, must submit, for
our review, any product that is not nationally known or that we have
not reviewed or used previously. Ads will not be accepted that do not indicate a telephone number.

Contact the 68 Micro Journal™ office for additional details.

Classified

Wanted to buy 'Dead or alive', 68XX devices. Computers, terminals, memory (etc) boards or anything else pertaining to 68XX computers or add-ons.

Please indicate condition and price in return. Give phone number where you can be reached.

Reply: Box 1, 68 Micro Journal™, 3018 Hamill Rd., Hixson, TN 37343

ISSUE DATE

Due to timing considerations of our publishing dates and the date on the cover of 68 Micro Journal™, we are dating this issue November/December 1979.

It WILL NOT affect how many issues you receive. If you are on a one year subscription you will receive 12 issues. If you are on a three year subscription you will receive a total of 36 issues. This means that your final issue will be dated one month later. In other words if you were supposed to expire in June 1980 (your subscription that is), your expiration (subscription) will be the July issue.

This will enable us to have your 68 Micro Journal™ to you prior to the cover date. As of now, because of printing and mailing considerations, we are being delivered late in the month of cover date. This we hope to remedy as we have found no way to force the postal service to deliver your magazine, as they claim they will. We find that in some instances it takes over three weeks for our own copy to be delivered to us, and we are right here where it is being mailed. I know of many readers in Texas and other distant points who recieve their 68 Micro Journal™ before we do. Even including Canada and Mexico, in some instances.

If your issues are being received late, contact your local postmaster and inform him (or her) that 68 Micro Journal™ is mailed under 2d class postal rates. This means that the delivery is "supposed" to be handled nearly as fast as 1st class and before all bulk or junk type mail.

If you do not receive any consideration from your local postmaster than let me know by letter and I will turn it over to the proper postal department. Maybe we can get things speeded up, or at least get what we are all paying for. Remember; 'The squeaking wheel gets greased first'!

******* CLASSIFIED ADVERTISING *******

'EXPANDOR' Black Box Printer Excellent Condition. $325.00 Postage Included. Paul Phelps, 111 Division St. No. 19, King City, Ca. 93930

SWTPC 6800 -12K of 2102 RAM-SWTBUG-Standard I/O with manuals $525.00 Call Larry Clark 1-615-842-7655

Smoke Signal BFD-68 or Percom LFD-400 Complete Single Drive Disk System. Includes Manuals, Diskettes. $500 Each Phil 1-312-653-3439

SWTPC CT-64 $200 Nenad Downing 4610 ST. Elmo Ave. Chatt, Tn. 37409 1-615-821-3727

HELP

Dear Sirs,

I have a SWTPC 6800, 12K byte,AC-30,CT-64, system. I have tried other sources (TSC, Cache) for information but without much success. Can you tell the following:

1) My system runs at 300 Baud and I wish to purchase a Worldwide Electronics Selectric Based Hard Copy Device. Can I run my CT-64 at 300 Baud while running the hard copy device at 110 provided I insert another serial interface into my computer?

2) Will the above mentioned hard copy device be compatible with my system?

3) I recently purchased TSC Text Editor and Text Processing Software. I can get the Text Editor to run OK, but the Text Processor comes up like the manual says, but won't accept Text. Do I have to have the Editor and Processor in memory at the same time in order to use the Text Processor?

Thank You,
Jeffery M. Craig
Apt. 912-3001 S. King dr
Chicago, Il. 60616

Dale Puckett
14753 Endsley
Woodbridge, VA 22193

A 68 MICRO JOURNAL LAB REVIEW

PASCAL

P-6800 PASCAL was written by David R. Gibby and the companion Run-Time System was created by Nigel W. Bennee. The package is published by Lucidata, Oosteinde 223, Voorburg 2271 EG, Netherlands.

P-6800 PASCAL is supplied on eitner a mini-FLEX or FLEX disk and comes with a 60-page user guide and reference manual. It costs $150.

In this review I will attempt to give you enough information to let you know what you can expect to receive if you purchase the package.

But first, we should introduce the language. PASCAL was created by Professor Niklaus Wirth of the Zurich Technical University. Wirth developed PASCAL so that he would have a language available which would allow him to teach programming as a systematic discipline.

Today PASCAL is one of the leading programming languages in computer science. Its forte is clarity.

Fundamental programming ideas can be expressed almost directly in PASCAL, making it a very easy language to read, understand and use. It is easy to understand because it combines the readability and general nature of the English language with the directness and preciseness of a machine language.

Gibby and Bennee designed the P-6800 compiler and run-time package specifically for the SWTP 6800 system. Because of a paging system which is invoked automaticaly when needed, it can compile and execute programs on a system with as litle as 12K of low memory plus the 4K or 8K required for FLEX. However, the more memory you have the faster the system will compile a program.

The compiler reads your source file and generates a file of pseudo or P-codes. These codes are interpreted later by the run-time system.

The P-codes are designed to be simple to decode, allowing programs to be executed tens of times faster than programs run with conventional interpreters, ie, BASIC. A program which has been compiled into its P-Code form is also very compact.

If you are skeptical of the power of the language you should know that the compiler itself is written in a subset of the language which it supports. Unfortunately however, the source of the compiler is not supplied.

DATA

There are two parts to a PASCAL program; DATA and STATEMENTS. DATA is declared or defined. It is held by a VARIABLE. It is declared by associating an identifier or name with a TYPE.

The TYPE limits the range of values of the DATA. TYPEs include BOOLEAN, CHAR, INTEGER, BYTE, SCALAR and ALFA. SCALAR types are defined by the programmer.

The DATA is structured into ARRAYS, SETS or FILES and all DATA in an ARRAY, SET or FILE must be of the same TYPE. This is a very important feature of PASCAL since it prevents you from accidently multiplying oranges by apples and winding up with nonsensical results.

BOOLEAN data has one of two values. It can either be true or false.

CHAR data may have the value of any character within the character set of your computer, ie, standard ASCII in most cases.

INTEGER data can be any whole number within the limits of the system. In this case it is plus or minus 32767.

BYTE is a special form of the data type INTEGER. It may have any value from 0 to 255.

Data of the type ALFA is an array of six characters which permits easy referencing of names and similar strings. It should be noted that standard PASCAL calls for an array of eight characters in ALFA.

STATEMENTS

The basic PASCAL STATEMENT is an ASSIGNMENT which evaluates an expression and calculates a new value for a variable. STATEMENTS may be compounded. If so, they contain several assignments grouped together between two delimiters named BEGIN and END. They may also be executed conditionally, using various programming constructs. Constructs avaialable in P-6800 PASCAL include: IF THEN ELSE, WHILE DO, REPEAT UNTIL, and FOR TO/DOWNTO DO.

PROCEDUREs and FUNCTIONs are statements that have been given a name. They may be used by a program, much like BASIC uses subroutines.

At this point we will list the various features that you receive in this version of PASCAL. Then, we will tell you which features are not available.

Reserved words in this implementation of PASCAL are: AND, ARRAY, BEGIN, CASE, CONST, DIV, DOWNTO, DO, ELSE, END, FILE, FOR, FUNCTION, IF, IN, NOT, OF, OR, PROCEDURE, PROGRAM, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, and WHILE.

Special symbols recognized by the compiler include:
+ * = , : ([(* " - / := ; .. .) *) < >
For readers not familiar with PASCAL, the (* symbol means open comment; *) means close comment; and the := means becomes.

On a negative note, you will find that the following standard reserved words are not found in this implementation: GOTO, LABEL, MOD, NIL, PACKED, RECORD, and WITH. Symbols not recognized by this version are the uparrow and the apostrophe or single quote.

Standard procedures and functions not available in P-6800 PASCAL include: PACK, UNPACK, NEW, PAGE, GET, PUT, ROUND, ABS, SQR, LN, ARCTAN, COS, SQRT, SIN, EXP, and TRUNC. The reason that most of these functions are missing is the fact that the REAL data type is not built into this implementation. POINTER and RECORD are also missing.

I feel that the omission of the REAL data type is the only major shortcoming of this software. Persons desiring to do a lot of math work are going to find the going difficult as they will have to devise various approximation algorithms, etc.

If I had enough time, I would sit down and interface my SWTPC calculator board to the compiler. Yet, if that condition were true, I would probably have enough time to write my own compiler and save $150. Again this is a major point, the price is pretty steep for an implementation that does not deal with REAL data.

On the positive side, there are several features implemented in P-6800 PASCAL that are not available in standard versions. They are: PEEK, POKE, HALT, USER and BYTE.

BYTE allows you to save memory when you know that your values always fall between 0 and 255. The others should be familiar to BASIC users. Obviously,they give you a tremendous amount of additional power and flexibility.

FLASH

After this review was written '68 Micro Journal™ received a letter from LUCIDATA describing a new version of P-6800. It will be called version 1.9 and will run in FLEX 2.0 or 6809 FLEX. It will allow the data types REAL and RECORDs and floating point numbers will have a nine digit precision.

And, here's more good news, LUCIDATA said that they would provide anyone who has purchased version 1.1 or 1.2 with an update for a "nominal charge" to cover the

cost of a disk and handling. They did not define the nominal charge.

SCALARS

A quick look at the enumerated TYPE is in order here, especially for the BASIC user who is not familiar with PASCAL. Here is an example:

```
TYPE
     PERSONS = (DALE, ESTHER, MICHELE);
     COLORS  = (RED, YELLOW, GREEN);
```

In the statement above a SET has been defined. The values are ordered according to the sequence in which they were listed in the definition.

Later in the program a SET type can be given a range of values, ie, GROUP = SET OF PERSONS or TRAFFICLIGHT = SET OF COLORS. I think these definitions quickly show you the power and clarity of this language.

FILE HANDLING

The following procedures are available in P-6800 PASCAL:

RESET (FILENAME). This statement will go to FLEX and CLOSE FILENAME, if it is open, then, OPEN FILENAME for read.

REWRITE (FILENAME) will close the file, DELETE any existing file with FILENAME and then OPEN a new file with the NAME, FILENAME. Obviously, extreme care must be taken when using these statements.

READLN (FILENAME) skips over the end of the current line to the first character of the next line. WRITELN (FILENAME) prepares to write the first character in the next line. READ (FILENAME, VALUE1, VALUE2, VALUEN) reads from the file and assigns the value read to the Variables in the list. WRITE (FILENAME, VALUE1, etc) writes the list to the proper file.

OPERATION

This software is easy to use. For instance, if you need to run a PASCAL program called DOG which you have already compiled, you only need type, RUN, DOG. FLEX will load and execute the run-time system which will execute the program DOG.

To compile a program the user types the source code into a .TXT file. He then uses the command line: RUN PASCAL PROGRAM.

When this command is received, the run-time system on the system disk is executed. It automatically looks for the compiler p-code on the system disk. After loading the compiler it searches for the text file on the work drive.

The compiler will then produce a binary p-code file and a listing file with the default extension .OUT on the work drive. It will use the system drive for a file called PROGRAM.SCR during compilation. All of the files are created and the drive assignments are made automatically by the run-time system.

OTHER COMMENTS

Overall, the manual supplied with P-6800 is detailed and adequate. It even provides a chapter that tells the user how to optimize his program so that the resulting p-code will be as efficient as possible. However, there are a few places in the manual that could stand improvement.

I was able to figure everything out by experimenting but I felt that specific examples of various operational commands should have been given in the manual. This could be done on one page in a chart which would contain a specific example of a command line for the many various options the user might require. For example, you can change the name of the listing file if you wish, or you can append a minus sign (-) to the end of the word PASCAL to compile a program without a listing.

The problem is that all of this information is buried in written text and is hard to find at a glance. A page of examples of various command lines would be a great help for the beginning user.

A confusing prompt in the program, combined with the sketchy operational details in the manual can really cause you to scratch your head for awhile. After the run-time system billboards itself, it asks you the question, "CHANGE STACK?" My first impression was that the machine wanted me to answer Yes or No, or maybe, Y or N. Not so.

After generating several error codes I discovered that I could enter a hex address without causing an error. It never even occurred to me for several days that I could have typed a carriage return to default to the present stack value of 1000 hex bytes. I think the manual should have included this detail.

On a positive note, it is possible to customize the run-time system and the manual provides the necessary information. Customizing is done by appending or overlaying user assembled object code to the RUN.CMD file. In fact, the authors provide two very useful overlays with the system. One allows you to execute PASCAL programs directly without answering the prompt for stack size changes. The other tells you what addresses to save to create a command file which will execute directly from FLEX.

Complete pictorial syntax diagrams are given for all statements in the implementation. Also eight simple programs are included on the disk which demonstrate to the user how to solve several problems with the language.

In conclusion, I feel that P-6800 PASCAL is pretty good for a first implementation. For users that do not require extensive math operations it provides an ideal way to get started in PASCAL. It is a tremendous improvement over BASIC.

Gibby and Bennee should be commended for taking the effort to make such a powerful language available to users of small systems with limited amounts of memory. With the addition of the data type REAL, random files and some higher math functions, plus just a little improvement in the documentation, this implementation would be outstanding.

Additional information can be secured by contacting:

LUCIDATA
Oosteinde 223
Voorburg 2271 EG, Netherlands

A 68 Micro Journal™ lab rating: AA

Rating Scale:
AAA - Excellent
AA - Good
A - Fair (could be better but works)
P - Poor (may not always work properly)
X - Not recommended for children
(or anything else!)

PSYMON (ASSY/SOURCE)

Percom Data Co.
211 N. Kirby
Garland, TX 75042

```
00001                           NAM   PSYMON
00004                   *****************************************************
00005                   * PSYMON VERSION 1.00                               *
00006                   *                                                   *
00007                   * COPYRIGHT (C) 1979 PERCOM DATA COMPANY, INC.      *
00008                   * PERCOM DATA COMPANY, INC. GRANTS UNRESTRICTED     *
00009                   * ROYALTY-FREE LICENSE FOR THE USE OF THIS          *
00010                   * PROGRAM PROVIDED THE USER CLEARLY ACKNOWLEDGES    *
00011                   * ITS ORIGIN.                                       *
00012                   *                                                   *
00013                   * WHILE THIS MONITOR IS VERY SIMPLE, ITS TRUE       *
00014                   * POWER LIES IN ITS EXTENSIBILITY AND IN THE        *
00015                   * TOOLS THAT IT PROVIDES FOR OTHER SOFTWARE         *
00016                   * TO USE.  THIS OPERATING SYSTEM IS DEDICATED       *
00017                   * TO HAROLD MAUCH  HO HIS LEGENDARY 512 BYTE        *
00018                   * OPERATING SYSTEM.                                 *
00019                   *                                                   *
00020                   * COMMANDS:                                         *
00021                   *   M <ADDRESS> - MEMORY EXAMINE/CHANGE             *
00022                   *   G <ADDRESS> - GO TO ADDRESS                     *
00023                   *   R <REGISTER> - REGISTER EXAMINE/CHANGE          *
00024                   *   L - LOAD PROGRAM FROM TAPE                      *
00025                   *   S <START> <END> - SAVE PROGRAM TO TAPE          *
00026                   *   B <ADDRESS> - SET/LIST BREAKPOINTS              *
00027                   *   U <ADDRESS> - UNSET BREAKPOINTS                 *
00028                   *   Z - JUMP TO PROM AT ADDRESS C000 H X            *
00029                   *                                                   *
00030                   * CALLABLE SUBROUTINES:                            *
00031                   *   INCHR - INPUT CHARACTER FROM CONSOLE            *
00032                   *   OUTCHR - OUTPUT CHARACTER TO CONSOLE            *
00033                   *   REQIO - PERFORM I/O TO PERIPHERAL               *
00034                   *   GETHEX - INPUT HEX NUMBER FROM CONSOLE          *
00035                   *   INHEX - INPUT HEX DIGIT FROM CONSOLE            *
00036                   *   DSPSBY - DISPLAY SINGLE BYTE & SPACE            *
00037                   *   DSPDBY - DISPLAY DOUBLE BYTE & SPACE            *
00038                   *   OUTHEX - DISPLAY 2 HEX DIGIST                   *
00039                   *   PSTRNG - DISPLAY STRING ON CONSOLE              *
00040                   *   LOAD - LOAD HEX PROGRAM FROM CONSOLE            *
00041                   *   SAVE - SAVE HEX PROGRAM TO CONSOLE              *
00042                   *   CRLF - BEGIN NEW LINE ON CONSOLE                *
00043                   *   OUTS - OUTPUT SPACE TO CONSOLE                  *
00044                   *                                                   *
00045                   * ALL I/O WITHIN PSYMON IS DONE THROUGH THE         *
00046                   * USE OF DEVICE CONTROL BLOCKS.  THIS ALLOWS        *
00047                   * EASY MODIFICATION BY THE USER.  PSYMON HAS        *
00048                   * FOUR DCB POINTERS INITIALIZED TO POINT TO THE     *
00049                   * CONSOLE (ACIA) DCB.  THEY ARE USED AS             *
00050                   * FOLLOWS:                                          *
00051                   *   CIDCB - POINTS TO DCB USED FOR CONSOLE          *
00052                   *           INPUT (CHARACTER I/O).                  *
00053                   *   CEDCB - POINTS TO DCB USED FO  ECHO OF          *
00054                   *           CHARACTERS RECEIVED USING CIDCB.        *
00055                   *           ECHO MAY BE SUPPRESSED BY SETTING       *
00056                   *           THIS POINTER TO ZERO.                   *
00057                   *   CODCB - POINTS TO DCB USED FOR CONSOLE          *
00058                   *           OUTPUT (CHARAC ER I/O).                 *
00059                   *   TPDCB - POINTS TO DCB USED FOR PSYMON           *
00060                   *           TAPE LOAD & SAVE COMMANDS.              *
00061                   *                                                   *
00062                   * THE PSYMON COMMAND TABLE  AY BE EXTENDED          *
00063                   * OR CHANGED BY SETTING THE POINTER 'USRTBL'        *
00064                   * TO THE ADDRESS OF A USER COMMAND TABLE.  IT       *
00065                   * IS INITIALIZED TO ZERO, INDICATING NO USER        *
00066                   * TABLE EXISTS.                                     *
00067                   *                                                   *
00068                   * THE PERCOM SYSTEM MONITOR (PSYMON) WAS            *
00069                   *  RITTEN BY A TEAM OF PROGRAMMERS U ING            *
00070                   * STRUCTURED TECHNIQUES.  THE TEAM MEMBERS          *
00071                   * ARE AS FOLLOWS:                                   *
00072                   *     HAROLD A MAUCH - PRESIDENT, PERCOM DATA       *
00073                   *     MIKE FOREMAN - 6809 PROJECT LEADER            *
00074                   *     BYRON SEASTRUNK - DESIGN ENGINEER             *
00075                   *     CLIFF RUSHING - PROGRAMMER                    *
00076                   *     JIM STUTSMAN - CHIEF PROGRAMMER               *
00077                   *                                                   *
00078                   * ADDITIONAL INFORMATION REGARDING THE USE OF       *
00079                   * 'PSYMON' MAY BE OBTAINED FROM:                    *
00080                   *     PERCOM DATA COMPANY, INC.                     *
00081                   *     211 NORTH KIRBY                               *
00082                   *     GARLAND, TEXAS  75042                         *
00083                   *                                                   *
00084                   *****************************************************
00086                   * SYSTEM ADDRESS CONSTANTS (MF-A2 VERSION)
00087 FC00              ROM1    EQU   $FC00     BASE ADD OF PSYMON ROM
00088 FE00              ROM2    EQU   $FE00     BASE ADD OF EXTENSION ROM
00089 A000              RAM     EQU   $A000     BASE ADD OF 128 BYTE RAM
00090 8004              TERMNL EQU    $8004     SYSTEM TERMINAL ACIA

00092                   * ASCII CHARACTER CONSTANTS
00093 000D              CR      EQU   $0D       CARRIAGE RETURN
00094 000A              LF      EQU   $0A       LINE FEED
00095 0020              SP      EQU   $20       SPACE
```

```
00097                          * ACIA CONTROL CONFIGURATIONS (MP-8 VERSION)
00098           0003   RESET   EQU   $03       RESET ACIA
00099           0011   CONFIG  EQU   $11       8 DATA, 2 STOP, NO PARITY
00100           0051   RDROW   EQU   CONFIG+$40  READER ON (RTS HIGH)
00101           0011   RDROFF  EQU   CONFIG    READER OFF (RTS LOW)

00103                          * PSYMON DCB OFFSETS
00104           0000   DCBLNK  EQU   0          PTR TO NEXT DCB IN CHAIN
00105           0002   DCBDID  EQU   2          ASCII 2 CHAR DEVICE ID
00106           0004   DCBDVR  EQU   4          DEVICE DRIVER ADDRESS
00107           0006   DCBIOA  EQU   6          DEVICE I/O ADDRESS
00108           0008    CBERR  EQU   8          ERROR STATUS CODE
00109           0009   DCBEXT  EQU   9          DCB EXTENSION BYTE COUNT
00110           000A   DCBAPP  EQU   10         DRIVER DCB APPENDAGE

00112                          * PSYMON DCB FUNCTION CODES
00113           0001   READFN  EQU   $01        READ FUNCTION CODE
00114           0002   WRITFN  EQU   $02        WRITE FUNCTION CODE
00115           0004   STATFN  EQU   $04        STATUS FUNCTION CODE
00116           0008   CNTLFN  EQU   $08        DEVICE CNTRL FUNCT CODE

00118                          * PSYMON RAM DEFINITIONS (MP-A2 VERSION)
00119 A000                     ORG   R

00121                          * PSYMON INTERNAL STACK (& REGISTER) SPACE
00122 A000      0028           RMB   40         STACK SPACE
00123           A028   STACK   EQU   *
00124 A028      0001   REGC    RMB   1          CONDITION CODE REGISTER
00125 A029      0001   REGA    RMB   1          A REGISTER
00126 A02A      0001   REGB    RMB   1          B REGISTER
00127 A02B      0001   REGD    RMB   1          DIRECT PAGE REGISTER
00128 A02C      0002   REGX    RMB   2          X REGISTER
00129 A02E      0002   REGY    RMB   2          Y REGISTER
00130 A030      0002   REGU    RMB   2          U STACK POINTER
00131 A032      0002   REGP    RMB   2          PROGRAM COUNTER

00133                          * PSYMON BREAKPOINT TABLE
00134 A034      001E   BPTABL  RMB   30         SPACE FOR 10 BREAKPOINTS
00135           A052   BPTEMD  EQU   *

00137                          * PSYMON WORK AREAS
00138 A052      0002   MEMPTR  RMB   2          MEMORY  POINTER FOR 'M' CMD
00139 A054      0002   USRTBL  RMB   2          ADDRESS OF USER COMMAND TBL
00140 A056      0001   COMAND  RMB   1          COMMAND CHAR STORAGE
00141 A057      0001   CKSUM   RMB   1          LOAD, SAVE CHECKSUM
00142 A058      0002   BEGADD  RMB   2          BEGIN ADDRESS FOR SAVE
00143 A05A      0002   ENDADD  RMB   2          END ADDRESS FOR SAVE
00144 A05C      0002   STKPTR  RMB   2          CONTENTS OF STACK PTR

00146                          * THE PSYM M CONSOLE DCB
00147 A05E      000A   CONDCB  RMB   10         NO EXTENSIONS

00149                          * PSYMON DCB POINTERS
00150 A068      0002   DCBCHN  RMB   2          BASE OF DCB CHAIN
00151 A06A      0002   CIDCB   RMB   2          CONSOLE INPUT DCB
00152 A06C      0002   CEDCB   RMB   2          CONSOLE ECHO DCB
00153 A06E      0002   CODCB   RMB   2          CONSOLE OUTPUT DCB
00154 A070      0002   TPDCB   RMB   2          CASSETTE TAPE DCB

00156                          * PSYMON VECTORS
00157 A072      0002   SWI3V   RMB   2          SOFTWARE INTERRUPT 3
00158 A074      0002   SWI2V   RMB   2          SOFTWARE INTERRUPT 2
00159 A076      0002   FIRQV   RMB   2          FAST INTERRUPT REQUEST
00160 A078      0002   IRQV    RMB   2          INTERRUPT REQUEST
00161 A07A      0002   SWIV    RMB   2          SOFTWARE INTERRUPT
00162 A07C      0002   NMIV    RMB   2          NON-MASKABLE INTERRUPT
00163 A07E      0002   RESTRT  RMB   2          RE-ENTRY INTO PSYMON
00165                          * PSYMON ROM CODING
00166 FC00                     ORG   ROM1
00167                          ********************************************
00168                          * PSYMON INITIALIZATION                    *
00169                          ********************************************
00170 FC00 10CE A028    4 INIT   LDS   #STACK    SET UP STACK POINTER
00171 FC04 1F   41      6        TFR   S,X        POINT X AT STACK
00172 FC06 6F   80      8 INIT1  CLR   ,X+        CLEAR A BYTE
00173 FC08 8C   A060    4        CMPX  #CONDCB+2  ALL FIELDS CLEAR?
00174 FC0B 26   F9      3        BNE   INIT1      LOOP IF NOT
00175 FC0D 10BE FFA5    4        LDY   #RAMINT    POINT TO RAM DATA
00176 FC11 EC   A1      8 INIT2  LDD   ,Y++       MOVE 2 BYTES
00177 FC13 ED   81      8        STD   ,X++
00178 FC15 8C   A080    4        CMPX  #RESTRT+2  END OF RAM?
00179 FC18 26   F7      3        BNE   INIT2      LOOP IF NOT
00180 FC1A 8E   A05E    3        LDX   #CONDCB    POINT TO DCB
00181 FC1D CC   0308    3        LDD   #RESET*256+CNTLFN  A=RESET, B=C TLFN
00182 FC20 BD   FD63    8        JSR   REQIO      RES T AC1A
00183 FC23 86   11      2        LDA   #CONFIG    CONFIGURE AC1A
00184 FC25 BD   FD63    8        JSR   REQIO
00185 FC28 B6   F800    5        LDA   ROM2       CHECK FOR SECOND ROM
00186 FC2B 81   7E      2        CMPA  #$7E       IS THERE A JUMP THERE?
00187 FC2D 26   03      3        BNE   MOMENT     GO IF NOT
00188 FC2F 80   F800    4        JSR   ROM2       CALL SECOND ROM

00190                          ********************************************
00191                          * PSYMON USER ENTRY                        *
00192                          ********************************************
00193 FC32 10CE A028    4 MOMENT LDS   #STACK    SET STACK POINTER
00194 FC36 10F7 A05C    7        STS   STKPTR

00196                          ********************************************
00197                          * GET COMMAND                              *
00198                          ********************************************
00199 FC3A 8E   FC4E    3 GETCMD LDX   #PROMPT   DISPLAY PROMPT
00200 FC3D BD   FD97    8        JSR   PSTRNG
00201 FC40 BD   FD44    8        JSR   INCHR      INPUT COMMAND CHARACTER
00202 FC43 8D   0F      7        BSR   LOOKUP     LOOK IT UP
00203 FC45 26   F3      3        BNE   GETCMD     LOOP IF NOT FOUND
00204 FC47 BD   FD75    8        JSR   OUTSP      OUTPUT A SPACE
00205 FC4A AD   94      10       JSR   [,X]       CALL COMMAND ROUTINE
00206 FC4C 20   8C      3        BRA   GETCMD     GO BACK FOR MORE

00208 FC4E      0D      PROMPT FCB   CR,LF
      FC4F      0A
00209 FC50      43             FCC   'CMD'
      FC51      4D
      FC52      44
00210 FC53      8F             FCB   '?+$80     END OF STRING

00212                          ********************************************
00213                          * LOOK UP COMMAND IN TABLE                 *
00214                          ********************************************
00215 FC54 108E A056    4 LOOKUP LDY   #COMAND   POINT Y TO COMMAND
00216 FC58 A7   A4      4        STA   ,Y        SAVE COMMAND CHAR CTER
00217 FC5A BE   A054    6        LDX   USRTBL     GET USER TABLE ADD ESS
00218 FC5D 27   04      3        BEQ   LOOK1      GO IF NONE
00219 FC5F 8D   05      7        BSR   SEARCH     SEARCH USER TABLE
00220 FC61 27   10      3        BEQ   SERCHX     GO IF FOUND
00221 FC63 8E   FF88    3 LOOK1  LDX   #CMDTBL    SEARCH INTERNAL TABLE

00223                          ********************************************
00224                          * GENERAL TABLE SEARCH                     *
00225                          *                                          *
00226                          * ENTRY REQUIREMENTS:  X - POINTS TO TABLE  *
00227                          *                      Y - POINTS TO ITEM  *
00228                          *                      FIRST BYTE OF TABLE MUST *
00229                          *                      CONTAIN ITEM LENGTH *
00230                          *                      LAST BYTE MU T BE FF *
00231                          *                                          *
00232                          * EXIT CONDITIONS:  C - Z SET IF FOUND, CLEAR *
00233                          *                       IF NOT FOUND       *
00234                          *                   X - POINTS TO ADDRESS OF *
00235                          *                       ROUTINE FOR MATCH  *
00236                          *                 A,B - CHANGED            *
00237                          *                                          *
00238                          ********************************************
00239 FC66 E6   80      6 SEARCH LDB   ,X+       GET ITEM LENGTH
00240 FC68 8D   0A      7 SERCH1 BSR   COMPAR    COMPARE CURRENT ITEM
00241 FC6A 3A           3        ABX             ADVANCE TO NEXT ITEM
00242 FC6B 27   06      3        BEQ   SERCHX     EXIT IF MATCH
00243 FC6D 30   02      5        LEAX  2,X        STE  OVER ADDRESS
00244 FC6F 6D   84      6        TST   ,X         END OF TABLE?
00245 FC71 2A   F5      3        BPL   SERCH1     LOOP IF NOT
00246 FC73 39           5 SERCHX RTS

00248                          ********************************************
00249                          * GENERAL STRING COMPARE                   *
00250                          *                                          *
00251                          * ENTRY REQUIREMENTS:  X - ADDRESS OF STRING 1 *
00252                          *                      Y - ADDRESS OF STRING 2 *
00253                          *                      B - LENGTH OF STRINGS *
00254                          *                                          *
00255                          * EXIT CONDITIO S:  C - SET PER COMPARE 1:2  *
00256                          *                   B,X,Y - UNCHANGED      *
00257                          *                   A - CHANGED            *
00258                          *                                          *
00259                          ********************************************
00260 FC74 34   34      9 COMPAR PSHS  B,X,Y     SAVE REGISTERS
00261 FC76 A6   80      6 COMP1  LDA   ,X+       GET NEXT CHARACTER
00262 FC78 A1   A0      6        CMPA  ,Y+       COMPARE IT
00263 FC7A 26   03      3        BNE   COMP2     EXIT IF UNMATCHED
00264 FC7C 5A           2        DECB            DECREMENT LOOP COUNT
00265 FC7D 26   F7      3        BNE   COMP1
00266 FC7F 35   84      11 COMP2 PULS  B,X,Y,PC RESTORE REGISTERS & EXIT

00268                          ********************************************
00269                          * LOAD PROGRAM FROM TAPE                   *
00270                          ********************************************
00271 FC81 FC   A06A    6 TLOAD  LDD   CIDCB     SAVE CONSOLE DCBS
00272 FC84 BE   A06C    6        LDX   CEDCB
00273 FC87 34   16      8        PSHS  A,B,X
00274 FC89 BE   A070    6        LDX   TPDCB      POINT TO TAPE DCB
00275 FC8C 4F           2        CLRA             SET D TO 0
00276 FC8D 5F           2        CLRB
00277 FC8E BF   A06A    6        STX   CIDCB      SET TAPE IN, NO ECHO
00278 FC91 FD   A06C    6        STD   CEDCB
00279 FC94 CC   5308    3        LDD   #RDROW*256+CNTLFN  RAISE RDR CNTRL
00280 FC97 8D   FD63    8        JSR   REQIO
00281 FC9A 8D   1B      7        BSR   LOAD       LOAD THE TAPE
00282 FC9C CC   1108    3        LDD   #RDROFF*256+CNTLFN  DROP R R CNTRL
00283 FC9F BE   A070    6        LDX   TPDCB
00284 FCA2 8D   FD63    8        JSR   REQIO
00285 FCA5 35   16      8        PULS  A,B,X      RESTORE CONSOLE DCBS
00286 FCA7 FD   A06A    6        STD   CIDCB
00287 FCAA BF   A06C    6        STX   CEDCB
00288 FCAD 7D   A057    7        TST   CKSUM      ANY ERRORS?
00289 FCB0 27   45      3        BEQ   LOADX      GO IF NOT

00291                          ********************************************
00292                          * DISPLAY ERROR INDICATOR OF '?'           *
00293                          ********************************************
00294 FCB2 86   3F      2 ERROR  LDA   #'?       DISPLAY ERROR INDICATOR
00295 FCB4 7E   FD58    4        JMP   OUTCHR

00297                          ********************************************
00298                          * LOAD PROGRAM IN HEX FORMAT               *
00299                          *                                          *
00300                          * ENTRY REQUIREMENTS:  NONE                *
00301                          *                                          *
00302                          * EXIT CONDITIONS:  ALL REGISTERS CHANGED  *
00303                          *                   CKSUM NON-ZERO IF ERROR *
00304                          *                                          *
00305                          ********************************************
00306 FCB7 1F   42      6 LOAD   TFR   S,Y       MARK STK FOR ERROR RCVRY
00307 FCB9 BD   FD44    8 LOAD1  JSR   INCHR     GET A CHARACTE
00308 FCBC 81   53      2 LOAD2  CMPA  #'S       START OF RECORD?
00309 FCBE 26   F9      3        BNE   LOAD1     LOOP IF NOT
00310 FCC0 BD   FD44    8        JSR   INCHR     GET ANOTHER CHARACTER
00311 FCC3 81   39      2        CMPA  #'9       END OF LOAD?
00312 FCC5 27   30      3        BEQ   LOADX     GO IF YES
00313 FCC7 81   31      2        CMPA  #'1       START OF RECORD?
00314 FCC9 26   F1      3        BNE   LOAD2     LOOP IF NOT
00315 FCCB 7F   A057    7        CLR   CKSUM     INIT CHECKSUM
00316 FCCE 8D   28      7        BSR   INBYTE    READ LENGTH
00317 FCD0 80   02      2        SUBA  #2        ADJUST IT
00318 FCD2 1F   89      6        TFR   A,B       SAVE IN B
00319 FCD4 8D   22      7        BSR   INBYTE    GET ADDRESS HI
00320 FCD6 A7   E3      7        STA   ,--S      SAVE ON STACK
00321 FCD8 8D   1E      7        BSR   INBYTE    GET ADDRESS LO
00322 FCDA A7   61      5        STA   1,S       PUT ON STACK
00323 FCDC 35   10      6        PULS  X         ADDRESS NOW IN X
00324 FCDE 8D   18      7 LOAD3  BSR   INBYTE    READ A BYTE
00325 FCE0 5A           2        DECB            DECREMENT COUNT
00326 FCE1 27   08      3        BEQ   LOAD4     GO IF DONE
00327 FCE3 A7   84      4        STA   ,X        STORE BYTE
00328 FCE5 A1   80      6        CMPA  ,X+       VERIFY GOOD ST RE
00329 FCE7 26   07      3        BNE   LOAD5     GO IF ERROR
```

```
00330 FCE9 20   F3       3          BRA   LOAD3
00331 FCEB 7C   A057     7 LOAD4    INC   CKSUM      CHECK CHECKSUM
00332 FCEE 27   C9       3          BEQ   LOAD1      LOOP IF GOOD
00333 FCF0 86   FF       2 LOAD5    LDA   #$FF       SET ERROR FLAG
00334 FCF2 B7   A057     5          STA   CKSUM
00335 FCF5 1F   24       6          TFR   Y,S        RESTORE STACK
00336 FCF7 39            5 LOADX    RTS

00338                                ********************************************
00339                                * INPUT BYTE                                *
00340                                ********************************************
00341 FCF8 8D   33       7 INBYTE   BSR   INHEX      GET HEX DIGIT
00342 FCFA 27   EF       3          BEQ   LOAD4      GO IF ERROR
00343 FCFC 48            2          ASLA             SHIFT TO MS HALF
00344 FCFD 48            2          ASLA
00345 FCFE 48            2          ASLA
00346 FCFF 48            2          ASLA
00347 FD00 34   02       5          PSHS  A          SAVE DIGIT
00348 FD02 8D   29       7          BSR   INHEX      GET ANOTHER DIGIT
00349 FD04 27   E5       3          BEQ   LOAD4      GO IF ERROR
00350 FD06 AB   E4       4          ADDA  ,S         COMBINE HALVES
00351 FD08 A7   84       4          STA   ,S         SAVE ON STACK
00352 FD0A BB   A057     5          ADDA  CKSUM      ADD TO CHECKSUM
00353 FD0D B7   A057     5          STA   CKSUM
00354 FD10 35   82       7          PULS  A,PC       GET RESULT & RETURN

00356                                ********************************************
00357                                * GET HEX NUMBER FROM CONSOLE                *
00358                                *                                            *
00359                                * ENTRY REQUIREMENTS:  NONE                  *
00360                                *                                            *
00361                                * EXIT CONDITIONS:  A - LAST CHAR INPUT      *
00362                                *                    B - HEX DIGIT COUNT     *
00363                                *                    X - HEX NUMBER          *
00364                                *                    C - SET ACCORDING TO B  *
00365                                *                                            *
00366                                ********************************************
00367 FD12 5F            2 GETHEX   CLRB             INIT DIGIT COUNT, RESULT
00368 FD13 8E   0000     3          LDX   #0
00369 FD16 8D   15       7 GETNX1   BSR   INHEX      GET A DIGIT
00370 FD18 27   11       3          BEQ   GETNX2     GO IF NOT HEX
00371 FD1A 1E   01       7          EXG   D,X        OLD RESULT TO A,B
00372 FD1C 58            2          ASLB             SHIFT LEFT 1 DIGIT
00373 FD1D 49            2          ROLA
00374 FD1E 58            2          ASLB
00375 FD1F 49            2          ROLA
00376 FD20 58            2          ASLB
00377 FD21 49            2          ROLA
00378 FD22 58            2          ASLB
00379 FD23 49            2          ROLA
00380 FD24 1E   01       7          EXG   D,X        REPLACE RESULT
00381 FD26 30   86       5          LEAX  A,X        ADD IN NEW DIGIT
00382 FD28 5C            2          INCB             ADD TO DIGIT COUNT
00383 FD29 20   E8       3          BRA   GETNX1     LOOP FOR MORE
00384 FD2B 5D            2 GETNX2   TSTB             SET/RESET Z FLAG
00385 FD2C 39            5          RTS

00387                                ********************************************
00388                                * GET HEX DIGIT FROM CONSOLE                 *
00389                                *                                            *
00390                                * ENTRY REQUIREMENTS:  NONE                  *
00391                                *                                            *
00392                                * EXIT CONDITIONS:  A - HEX DIGIT OR NON-HEX *
00393                                *                    C - Z FLAG SET IF A NOT HEX *
00394                                *                    ALL OTHER REGS PRESERVED *
00395                                *                                            *
00396                                ********************************************
00397 FD2D 8D   15       7 INHEX    BSR   INCHNP     GET A CHARACTER
00398 FD2F 34   02       5          PSHS  A          SAVE IT
00399 FD31 80   30       2          SUBA  #$30       CONVERT TO BINARY
00400 FD33 2B   0A       3          BMI   INHEX2     GO IF NOT NUMERIC
00401 FD35 81   09       2          CMPA  #$09       GREATER THAN 9?
00402 FD37 23   02       3          BLS   INHEX1     GO IF NOT
00403 FD39 80   07       2          SUBA  #$07       CONVERT LETTER
00404 FD3B 81   0F       2 INHEX1   CMPA  #$0F       GREATER THAN 15?
00405 FD3D 23   02       3          BLS   INHEX3     GO IF NOT
00406 FD3F A6   E4       4 INHEX2   LDA   ,S         GET ORIGINAL CHAR BACK
00407 FD41 A1   E0       6 INHEX3   CMPA  ,S+        SET/RESET Z FLAG
00408 FD43 39            5          RTS

00410                                ********************************************
00411                                * CONSOLE INPUT ROUTINE                      *
00412                                *                                            *
00413                                * ENTRY REQUIREMENTS:  NONE                  *
00414                                *                                            *
00415                                * EXIT CONDITIONS:  A - CHARACTER WITH PARITY *
00416                                *                      REMOVED               *
00417                                *                    ALL OTHER REGS PRESERVED *
00418                                *                    EXCEPT C                *
00419                                *                                            *
00420                                ********************************************
00421 FD44 34   14       7 INCHNP   PSHS  B,X        SAVE REGISTERS
00422 FD46 8E   A06A     6          LDX   CIODCB     POINT TO INPUT DCB
00423 FD49 C6   01       2          LDB   #READFN    SET UP FOR READ
00424 FD4B 8D   16       7          BSR   REQIO      READ A CHARACTER
00425 FD4D 84   7F       2          ANDA  #$7F       REMOVE PARITY
00426 FD4F 8E   A06C     6          LDX   CEDCB      POINT TO ECHO DCB
00427 FD52 34   02       5          PSHS  A          SAVE CHARACTER
00428 FD54 26   07       3          BNE   OUTCH1     GO IF ECHO
00429 FD56 35   96       10         PULS  A,B,X,PC RESTORE & RETURN

00431                                ********************************************
00432                                * CONSOLE OUTPUT ROUTINE                     *
00433                                *                                            *
00434                                * ENTRY REQUIREMENTS:  A - CHARACTER TO BE   *
00435                                *                          OUTPUT TO CONSOLE *
00436                                *                                            *
00437                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00438                                *                    EXCEPT C                *
00439                                *                                            *
00440                                ********************************************
00441 FD 8 34   16       8 OUTCHR   PSHS  A,B,X      SAVE REGISTERS
00442 FD5A BE   A06E     6          LDX   CODCB      POINT TO OUTPUT DCB
00443 FD5D C6   02       2 OUTCH1   LDB   #WRITFN    SET FUNCTION
00444 FD5F 8D   02       7          BSR   REQIO      OUTPUT THE CHARACTER
00445 FD61 35   96       10         PULS  A,B,X,PC RESTORE REGISTERS & RETURN

00447                                ********************************************
00448                                * PERFORM I/O REQUESTS                       *
00449                                *                                            *
00450                                * ENTRY REQUIREMENTS:  A - DRIVER PARAMETER  *
00451                                *                      B - FUNCTION CODE     *
00452                                *                      X - DCB ADDRESS       *
00453                                *                                            *
00454                                * EXIT CONDITIONS:  A - DRIVER RESULT        *
00455                                *                    ALL OTHERS PRESERVED    *
00456                                *                    EXCEPT C                *
00457                                *                                            *
00458                                ********************************************
00459 FD63 34   34       9 REQIO    PSHS  B,X,Y      SAVE REGISTERS
00460 FD65 AD   98 04    12         JSR   [DCBDVR,X]  CALL DRIVER
00461 FD68 35   84       11         PULS  B,X,Y,PC RESTORE REGISTERS & EXIT

00463                                ********************************************
00464                                * DISPLAY DOUBLE BYTE                        *
00465                                *                                            *
00466                                * ENTRY REQUIREMENTS:  A,B - DOUBLE BYTE     *
00467                                *                            TO BE PRINTED   *
00468                                *                                            *
00469                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00470                                *                    EXCEPT C                *
00471                                *                                            *
00472                                ********************************************
00473 FD6A 8D   11       7 DSPDBY   BSR   OUTHEX     DISPLAY A AS 2 HEX DIGITS
00474 FD6C 1E   89       7          EXG   A,B        LS BYTE TO A
00475 FD6E 8D   01       7          BSR   DSPBBY     DISPLAY AS 2 DIGITS, SPACE
00476 FD70 1E   89       7          EXG   A,B        RESTORE A & B
00477 FD72 39            5          RTS

00479                                ********************************************
00480                                * DISPLAY A BYTE AND SPACE                   *
00481                                *                                            *
00482                                * ENTRY REQUIREMENTS:  A - BYTE TO BE DISPLAYED *
00483                                *                                            *
00484                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00485                                *                    EXCEPT C                *
00486                                *                                            *
00487                                ********************************************
00488 FD73 8D   08       7 DSPBBY   BSR   OUTHEX     DISPLAY BYTE IN A

00490                                ********************************************
00491                                * OUTPUT A SPACE TO THE CONSOLE              *
00492                                *                                            *
00493                                * ENTRY REQUIREMENTS:  NONE                  *
00494                                *                                            *
00495                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00496                                *                    EXCEPT C                *
00497                                *                                            *
00498                                ********************************************
00499 FD75 34   02       5 OUTSP    PSHS  A          SAVE A REGISTER
00500 FD77 86   20       2          LDA   #SP        OUTPUT A SPACE

00502                                ********************************************
00503                                * OUTPUT CHARACTER, RESTORE A, & RETURN      *
00504                                ********************************************
00505 FD79 8D   DD       7 OUTCHR   BSR   OUTCHR     DISPLAY CHARACTER
00506 FD7B 35   82       7          PULS  A,PC       RESTORE & EXIT

00508                                ********************************************
00509                                * DISPLAY A REGISTER AS 2 HEX DIGITS         *
00510                                *                                            *
00511                                * ENTRY REQUIREMENTS:  A - BYTE TO DISPLAY   *
00512                                *                                            *
00513                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00514                                *                    EXCEPT C                *
00515                                *                                            *
00516                                ********************************************
00517 FD7D 34   02       5 OUTHEX   PSHS  A          SAVE THE BYTE
00518 FD7F 44            2          LSRA             GET MS DIGIT
00519 FD80 44            2          LSRA
00520 FD81 44            2          LSRA
00521 FD82 44            2          LSRA
00522 FD83 8D   06       7          BSR   OUTDIG     DISPLAY IT
00523 FD85 A6   E4       4          LDA   ,S         GET LS DIGIT
00524 FD87 8D   02       7          BSR   OUTDIG     DISPLAY IT
00525 FD89 35   82       7          PULS  A,PC       RESTORE A & RETURN

00527                                ********************************************
00528                                * DISPLAY A HEX DIGIT                        *
00529                                ********************************************
00530 FD8B 84   0F       2 OUTDIG   ANDA  #$0F       MASK OFF DIGIT
00531 FD8D 8B   30       2          ADDA  #$30       CONVERT TO ASCII
00532 FD8F 81   39       2          CMPA  #$39       BIGGER THAN 9?
00533 FD91 23   C5       3          BLS   OUTCHR     GO IF NOT
00534 FD93 8B   07       2          ADDA  #$07       CONVERT TO LETTER
00535 FD95 20   C1       3          BRA   OUTCHR     PRINT AND EXIT

00537                                ********************************************
00538                                * PRINT A STRING TO THE CONSOLE              *
00539                                *                                            *
00540                                * ENTRY CONDITIONS:  X - POINTS TO STRING    *
00541                                *                     LAST BYTE HAS BIT 7 ON *
00542                                *                                            *
00543                                * EXIT CONDITIONS:  X - POINTS 1 BYTE PAST END *
00544                                *                     A,C - CHANGED          *
00545                                *                                            *
00546                                ********************************************
00547 FD97 A6   84       4 PSTRNG   LDA   ,X         GET A CHARACTER
00548 FD99 84   7F       2          ANDA  #$7F       MASK OFF
00549 FD9B 8D   BB       7          BSR   OUTCHR     DISPLAY IT
00550 FD9D 6D   80       8          TST   ,X+        WAS IT LAST?
00551 FD9F 2A   F6       3          BPL   PSTRNG     LOOP IF NOT
00552 FDA1 39            5          RTS

00554                                ********************************************
00555                                * PRINT CR/LF ON CONSOLE                     *
00556                                *                                            *
00557                                * ENTRY REQUIREMENTS:  NONE                  *
00558                                *                                            *
00559                                * EXIT CONDITIONS:  ALL REGISTERS PRESERVED  *
00560                                *                    EXCEPT C                *
00561                                *                                            *
00562                                ********************************************
00563 FDA2 34   02       5 CRLF     PSHS  A          SAVE A REGISTER
00564 FDA4 86   0D       2          LDA   #CR        OUTPUT CR
00565 FDA6 8D   B0       7          BSR   OUTCHR
00566 FDA8 86   0A       2          LDA   #LF        OUTPUT LF & EXIT
00567 FDAA 20   CD       3          BRA   OUTCSR
```

```
                        ********************************************************
                        * SAVE PROGRAM ON TAPE                                 *
                        ********************************************************
00572 FDAC 8D   30     7 TSAVE  BSR   GETHX    GET START ADDRESS
00573 FDAE 27   0E     3        BEQ   TSAVE2   GO IF NONE
00574 FDB0 BF   A05B   6        STX   BEGADD   SAVE START
00575 FDB3 8D   29     7        BSR   GETHX    GET END ADDRESS
00576 FDB5 26   04     3        BNE   TSAVE1   GO IF ENTERED
00577 FDB7 BE   A05B   6        LDX   BEGADD   DUPLICATE ADDRESS
00578 FDBA 5C          2        INCB           SET ADDRESS INDICATOR
00579 FDBB BF   A05A   6 TSAVE1 STX   ENDADD   SAVE END
00580 FDBE BE   A06E   6 TSAVE2 LDX   CODCB    SAVE CONSOLE DCB
00581 FDC1 34   12     7        PSHS  A,X      SAVE TERMINATOR TOO
00582 FDC3 BE   A070   6        LDX   TPDCB    SET UP FOR TAPE
00583 FDC6 BF   A06E   6        STX   CO  B
00584 FDC9 5D          2        TSTB           ANY ADDRESS ENTERED?
00585 FDCA 27   02     3        BEQ   TSAVE3   GO IF NOT
00586 FDCC 8D   13     7        BSR   SAVE     SAVE THE PROGRAM
00587 FDCE 35   02     5 TSAVE3 PULS  A        GET TERMINATOR
00588 FDD0 81   0D     2        CMPA  #CR      WAS IT RETURN?
00589 FDD2 26   04     3        BNE   TSAVE4   DO IF NOT
00590 FDD4 C6   39     2        LDB   #'9      OUTPUT S9 RECORD
00591 FDD6 8D   54     7        BSR   OUTSN
00592 FDD8 35   10     6 TSAVE4 PULS  X        RESTORE DCB POI TER
00593 FDDA BF   A06E   6        STX   CODCB
00594 FDDD 39          5        RTS

                        ********************************************************
                        * GET HEX NUMBER IN X                                  *
                        ********************************************************
00599 FDDE 7E   FD12   4 GETHX  JMP   GE HEX   RELATIVE BRANCH BOOSTER

                        ********************************************************
                        * SAVE A PROGRA  IN HEX                                *
                        *                                                      *
                        * ENTRY REQUIREMENTS:  SAVE ADDRESSES ARE IN           *
                        *                        BEGADDR & ENDADDR             *
                        *                                                      *
                        * EXIT CONDITIONS:  ALL REGISTERS C  NGED              *
                        *                                                      *
                        ********************************************************
00610 FDE1 BE   A05B   6 SAVE   LDX   BEG DD   POINT AT FIRST BYTE
00611 FDE4 C6   31     2 SAVE1  LDB   #'1      BEGIN NEW S1 RECORD
00612 FDE6 8D   44     7        BSR   OUTSN
00613 FDE8 7F   A057   7        CLR   CKSUM    INIT CHECKSUM
00614 FDEB FC   A05A   6        LDD   ENDADD   CALCULATE BYTES TO SAVE
00615 FDEE 34   10     6        PSHS  X
00616 FDF0 A3   E1     9        SUBD  ,S++
00617 FDF2 4D          2        TSTA           GREATER THAN 255?
00618 FDF3 26   04     3        BNE   SAVE2    DO IF YES
00619 FDF5 C1   10     2        CMPB  #16      LESS THAN FULL RECORD?
00620 FDF7 25   02     3        BLO   SAVE3    GO IF YES
00621 FDF9 C6   0F     2 SAVE2  LDB   #15      SET FULL RECORD SIZE
00622 FDFB 5C          2 SAVE3  INCB           CORRECT REC RD SIZE
00623 FDFC 1F   98     6        TFR   B,A      OUTPUT RECORD SIZE
00624 FDFE 8B   03     2        ADDA  #3       ADJUST FOR ADDRESS,COUNT
00625 FE00 8D   20     7        BSR   OUTBYT
00626 FE02 34   10     6        PSHS  X        ADDRESS TO STACK
00627 FE04 35   02     5        PULS  A        OUTPUT ADDRESS HI
00628 FE06 8D   1A     7        BSR    UTBYT
00629 FE08 35   02     5        PU S  A        OUTPUT ADDRESS LO
00630 FE0A 8D   16     7        BSR   OUTBY
00631 FE0C A6   80     6 SAVE4  LDA   ,X+      SAVE A DATA BYTE
00632 FE0E 8D   12     7        BSR   OUTBYT
00633 FE10 5A          2        DECB           LOOP UNTIL 0
00634 FE11 25   F9     3        BNE   SAVE4
00635 FE13 B6   A057   5        LDA   CKSUM    GET CHECKSUM
00636 FE16 43          2        COMA           COMPLIMENT IT
00637 FE17 8D   09     7        BSR   OUTBYT   OUTPUT IT
00638 FE19 31   1F     5        LEAY  -1,X     CHECK FOR END
00639 FE1B 10BC A05A   8        CMPY  ENDADD
00640 FE1F 26   C3     3        BNE   SAVE1    LOOP IF NOT
00641 FE21 39          5        RTS

                        ********************************************************
                        * OUTPUT BYTE AS HEX AND ADD TO CHECKSUM               *
                        ********************************************************
00646 FE22 8D   FD70   8 OUTBYT JSR   OUTHEX   OUTPUT BYTE AS HEX
00647 FE25 BB   A057   5        ADDA  CKSUM    ADD TO CHECKSUM
00648 FE28 B7   A057   5        STA   CKSUM
00649 FE2B 39          5        RTS

                        ********************************************************
                        * OUTPUT 'S' TAPE RECORD HEADERS                       *
                        ********************************************************
00654 FE2C 8D   FDA2   8 OUTSN  JSR   CRLF     BEGIN NEW LINE
00655 FE2F 86   53     2        LDA   #'S      OUTPUT 'S' HEADER
00656 FE31 8D   02     7        BSR    UTC
00657 FE33 1F   98     6        TFR   B,A      RECORD TYPE TO A

                        ********************************************************
                        * OUTPUT CHARACTER TO CONSOLE                          *
                        ********************************************************
00662 FE35 7E   FD58   4 OUTC   JMP   OUTCHR   RELATIVE BRANCH BOOSTER

                        ********************************************************
                        * MEMORY EXAMINE AND CHANGE                            *
                        ********************************************************
00667 FE38 8D   A4     7 MEMEC  BSR   GETHX    GET ADDRESS
00668 FE3A 26   03     3        BNE   MEMEC1   GO IF GOOD
00669 FE3C BE   A052   6        LDX   MEMPTR   USE PREVIOUS
00670 FE3F BF   A052   6 MEMEC1 STX   MEMPTR   UPDATE RAM POINTER
00671 FE42 BD   F0A2   8        JSR   CRLF     BEGIN NEW LINE
00672 FE45 1F   10     6        TFR   X,D      DISPLAY ADDRESS
00673 FE47 BD   FD6A   8        JSR   DSPBBY
00674 FE4A A6   80     6        LDA   ,X+      GET CONTENTS
00675 FE4C BD   FD73   8        JSR   DSPSBY   DISPLAY THEM
00676 FE4F 1F   12     6        TFR   X,Y      SAVE ADDRESS IN Y
00677 FE51 8D   88     7        BSR   GETHX    GET CHANGE DATA
00678 FE53 1E   01     7        EXG   D,X      SAVE DELIM, GET NEW
00679 FE55 27   09     3        BEQ   M MSC2   GO IF NO CHANGE
00680 FE57 E7   3F     5        STB   -1,Y     UPDATE MEMORY
00681 FE59 E1   3F     5        CMPB  -1,Y     VERIFY GOOD STORE
00682 FE5B 27   03     3        BEQ   MEMEC2   GO IF GOOD STORE
00683 FE5D BD   FCB2   8        JSR   ERROR    DISPLAY ERROR
00684 FE60 1F   10     6 MEMEC2 TFR   X,D      GET DELIMITER IN A
00685 FE62 1F   21     6        TFR   Y,X      GET NEXT ADDRESS IN X
00686 FE64 81   0D     2        CMPA  #CR      END OF UPDATE?
00687 FE66 27   08     3        BEQ   MEMEC3   DO IF YES
00688 FE68 81   5E     2        CMPA  #'^      BACKING UP?
00689 FE6A 26   D3     3        BNE   MEMEC1   LOOP IF NOT
00690 FE6C 30   83     7        LEAX  ,--X     BACK UP 2
00691 FE6E 20   CF     3        BRA   MEMEC1   CONTINUE
00692 FE70 39          5 MEMEC3 RTS

                        ********************************************************
                        * GO TO ADDRESS                                        *
                        ********************************************************
00697 FE71 10F  A05C   7 GO     LDS   STKPTR   SET UP STACK
00698 FE75 BD   FD12   8        JSR   GETHEX   GET TARGET ADDRESS
00699 FE78 27   02     3        BEQ   Go1      GO IF NONE
00700 FE7A AF   6A     6        STX   10,S     STORE IN PC ON STACK
00701 FE7C A6   E4     4 Go1    LDA   ,S       SET 'E' FLAG IN CC
00702 FE7E 8A   80     2        ORA   #$80
00703 FE80 A7   84     4        STA   ,S
00704 FE82 3B         15 INTRET RTI            LOAD REGISTERS AND GO

                        ********************************************************
                        * BREAKPOINT (SOFTWARE INTERRUPT) TRAP                 *
                        ********************************************************
00709 FE83 AE   6A     6 BREPNT LDX   10,S     GET PROGRAM COUNTER
00710 FE85 30   1F     5        LEAX  -1,X     DECREMENT BY 1
00711 FE87 AF   6A     6        STX   10,S     REPLACE ON STACK
00712 FE89 C6   FF     2        LDB   #$FF     FLAG FOR SINGLE REMOVAL
00713 FE8B BD   FF43   8        JSR   REMBK    REMOVE BREAKPOINT

                        ********************************************************
                        * INTERRUPT (HARDWARE/SOFTWARE) TRAP                   *
                        ********************************************************
00718 FE8E 10F7 A05C   7 TRAP   STS   STKPTR   SAVE STACK POINTER
00719 FE92 BD   FDA2   8        JSR   CRLF     BEGIN NEW LINE
00720 FE95 8D   3C     7        BSR   REGDMP   DUMP REGISTERS
00721 FE97 7E   FC3A   4        JMP   GETCMD   GET NEXT COMMAND

                        ********************************************************
                        * REGISTER EXAMINE AND CHANGE                          *
                        ********************************************************
00726 FE9A 8D   FD44   8 REGEC  JSR   INCHR    GET REGISTER  O EXAMINE
00727 FE9D 8D   F0A2   8        JBR   CRLF     BEGIN NEW LINE
00728 FEA0 5F          2        CLRB           CLEAR OFFSET COUNT
00729 FEA1 8E   FEC7   3        LDX   #REGIDS  POINT TO REGISTER ID STRING
00730 FEA4 A1   85     5 REGEC1 CMPA  B,X      CHECK REGISTER NAME
00731 FEA6 27   07     3        BEQ   REGEC2   GO IF FOUND
00732 FEA8 5C          2        INCB           ADVANCE COUNTER
00733 FEA9 C1   0B     2        CMPB  #11      END OF LIST?
00734 FEAB 23   F7     3        BLS   REGEC1   LOOP IF NOT
00735 FEAD 20   24     3        BRA   REGDMP   BAD ID - DUMP ALL
00736 FEAF 34   04     5 REGEC2 PSHS  B        SAVE OFFSET
00737 FEB1 8D   37     7        BSR   RDUMP    DISPLAY THE REG & CONTENTS
00738 FEB3 8D   FD12   8        JSR   GETHEX   GET NEW VALUE
00739 FEB6 35   04     5        PUL   B        RESTORE OFFSET
00740 FEB8 27   0C     3        BEQ   REGBCX   GO IF NO CHANGE
00741 FEBA 31   A5     5        LEAY  B,Y      POINT TO REG ON STACK
00742 FEBC C1   03     2        CMPB  #3       SINGLE BYTE REG?
00743 FEBE 1F   10     6        TFR   X,D      GET NEW DATA IN A,B
00744 FEC0 23   02     3        BLS   REGEC3   DO IF SIN LE
00745 FEC2 A7   A0     6        STA   ,Y+      STORE MS BYTE
00746 FEC4 E7   A4     4 REGEC3 STB   ,Y       S ORE LS BYTE
00747 FEC6 39          5 REGBCX RTS

00749 FEC7 43          REGIDS FCC   'CABDXXYYUUPP'
       FEC8 41
       FEC9 42
       FECA 44
       FECB 58
       FECC 58
       FECD 59
       FECE 59
       FECF 55
       FED0 55
       FED1 50
       FED2 50

                        ********************************************************
                        * COMPLETE REGISTER DUMP                               *
                        ********************************************************
00754 FED3 8E   FEC7   3 REGDMP LDX   #REGIDS  POINT TO ID STRING
00755 FED6 5F          2        CLRB           CLEAR OFFSET COUNTER
00756 FED7 A6   85     5 RGDMP1 LDA   B,X      GET REG NAME
00757 FED9 8D   07     7        BSR   RDUMP    DISPLAY IT
00758 FEDB 5C          2        INCB           BUMP TO NEXT REG
00759 FEDC C1   0B     2        CMPB  #11      ALL PRINTED?
00760 FEDE 23   F7     3        BLS   RGDMP1   LOOP IF NOT
00761 FEE0 86   53     2        LDA   #'S      DISPLAY STACK ID
00762 FEE2 8D   18     7        BSR   DSPID
00763 FEE4 108E A050   4        LDY   #STKPTR-12  Y+B=>STKPTR
00764 FEE8 20   0A     3        BRA   RDUMP1

                        ********************************************************
                        * DISPLAY REGISTER CONTENTS                            *
                        ********************************************************
00769 FEEA 8D   13     7 RDUMP  BSR   DSPID    DISPLAY REGISTER ID
00770 FEEC 108E A05C   7        LDY   STKPTR   POINT Y AT STACK
00771 FEF0 C1   03     2        CMPB  #3       SINGLE BYTE REG?
00772 FEF2 23   06     3        BLS   RDUMP2   GO IF YES
00773 FEF4 A6   A5     5 RDUMP1 LDA   B,Y      DISPLAY MS BYTE
00774 FEF6 8D   FD70   8        JSR   OUTHEX
00775 FEF9 5C          2        I CB           ADVANCE OFFSET
00776 FEFA A6   A5     5 RDUMP2 LDA   B,Y      DISPLAY A BYTE
00777 FEFC 7E   F073   4        JMP   DSPBBY

                        ********************************************************
                        * DISPLAY REGISTER ID                                  *
                        ********************************************************
00782 FEFF 8D   02     7 DSPID  BSR   OUTCH    DISPLAY REG NAME
00783 FF01 86   3D     2        LDA   #'=      DISPLAY '='

                        ********************************************************
                        * OUTPUT CHARACTER TO CONSOLE                          *
                        ********************************************************
00788 FF03 7E   FD58   4 OUTCH  JMP   OUTCHR   RELATIVE BRANCH BOOSTER

                        ********************************************************
                        * SET A BREAKPOINT                                     *
                        ********************************************************
00793 FF06 BD   FD12   8 SETBK  JSR   GETHEX   GET ADDRESS
00794 FF09 27   18     3        BEQ   DSPBX    GO IF NONE ENTERED
00795 FF0B 80   27     7        BSR   INTBP    POINT Y AT BP TAB E
```

```
00796 FF0D EC   A4       5 SETBK1 LDD   ,Y        EMPTY SLOT?
00797 FF0F 27   06       3        BEQ   SETBK2     GO IF YES
00798 FF11 8D   26       7        BSR   NEXTBP     ADVANCE TO NEXT SLOT
00799 FF13 26   F8       3        BNE   SETBK1     LOOP IF NOT END
00800 FF15 20   0C       3        BRA   EXIT       EXIT
00801 FF17 AF   A4       5 SETBK2 STX   ,Y        SAVE ADDRESS
00802 FF19 27   08       3        BEQ   DSPBK      GO IF ADDRESS = 0
00803 FF1B A6   84       4        LDA   ,X         GET CONTENTS
00804 FF1D A7   22       5        STA   2,Y        SA E IN TABLE
00805 FF1F 86   3F       2        LDA   #$3F       SWI OP C  E
00806 FF21 A7   84       4        STA   ,X         SET BREAK

00808                               ********************************************
00809                               * DISPLAY ALL BREAKPOINTS                  *
00810                               ********************************************
00811 FF23 BD   FDA2     8 DSPBK  JSR   CRLF      BEGIN NEW LINE
00812 FF26 8D   0C       7        BSR   INITBP     POINT Y AT BP TABLE
00813 FF28 EC   A4       5 DSPBK1 LDD   ,Y        GET ADDRESS OF BP
00814 FF2A 27   03       3        BEQ   DSPBK2     GO IF INACTIVE
00815 FF2C BD   FD6A     8        JSR   DSPDBY     DISPLAY ADDRESS
00816 FF2F BD   08       7 DSPBK2 BSR   NEXTBP     ADVANCE POINTER
00817 FF31 26   F5       3        BNE   DSPBK1     LOOP IF NOT END
00818 FF33 39            5        RTS

00820                               ********************************************
00821                               * INITIALIZE BREAKPOINT TABLE POINTER      *
00822                               ********************************************
00823 FF34 108E A034     4 INITBP LDY   #BPTABL   POINT Y AT BP TABLE
00824 FF38 39            5        RTS

00826                               ********************************************
00827                               * ADVANCE BREAKPOINT TABLE POINTER         *
00828                               ********************************************
00829 FF39 31   23       5 NEXTBP LEAY  3,Y       ADVANCE TO NEXT ENTRY
00830 FF3B 108C A052     5        CMPY  #BPTEND   CHECK FOR END OF TABLE
00831 FF3F 39            5        RTS

00833                               ********************************************
00834                               * UNSET A BREAKPOINT                       *
00835                               ********************************************
00836 FF40 BD   FDE2     8 UNSBK  JSR   GETHEX    GET ADDRESS

00838                               ********************************************
00839                               * REMOVE ONE OR MORE BREAKPOINTS           *
00840                               ********************************************
00841 FF43 BD   FF       7 REMBK  BSR   INITBP    POINT Y AT BP TABLE
00842 FF45 5D            2 REMBK1 TSTB            REMOVE ALL?
00843 FF46 27   06       3        BEQ   REMBK2    GO IF YES
00844 FF48 AC   A4       6        CMPX  ,Y        FIND ADDRESS?
00845 FF4A 27   09       3        BEQ   UNSET     GO IF YES
00846 FF4C 20   02       3        BRA   REMBK3    LOOP IF NO
00847 FF4E 8D   05       7 REMBK2 BSR   UNSET     UNSET IT
00848 FF50 8D   87       7 REMBK3 BSR   NEXTBP    ADVANCE POINTER
00849 FF52 26   F1       3        BNE   REMBK1    LOOP IF NOT END
00850 FF54 39            5        RTS

00852                               ********************************************
00853                               * REMOVE A BREAKPOINT                      *
00854                               ********************************************
00855 FF55 AE   A4       5 UNSET  LDX   ,Y        GET ADDRESS OF BP
00856 FF57 27   08       3        BEQ   UNSET1    GO IF INACTIVE
00857 FF59 A6   22       5        LDA   2,Y       GET CONTENTS
00858 FF5B A7   84       4        STA   ,X        REPLACE BP
00859 FF5D 6F   A4       6        CLR   0,Y       MARK BP INACTIVE
00860 FF5F 6F   21       7        CLR   1,Y
00861 FF61 39            5 UNSET1 RTS

00863                               ********************************************
00864                               * TERMINAL DRIVER (ACIA)                   *
00865                               ********************************************
00866 FF62 6F   08       7 TERMOR CLR   DCBERR,X  NO ERRORS POSSIBLE
00867 FF64 AE   06       6        LDX   DCBIOA,X  GET I/O ADDRESS
00868 FF66 54            2        LSRB             READ FUNCTION?
00869 FF67 25   0C       3        BCS   TERMRD    GO IF YES
00870 FF69 54            2        LSRB             WRIT  FU  TION?
00871 FF6A 25   11       3        BCS   TERMWT    GO IF YES
00872 FF6C 54            2        LSRB             STATUS FUNCTION?
00873 FF6D 25   17       3        BCS   TER ST    GO IF YES
00874 FF6F 54            2        LSRB             CONTROL FUNCTION?
00875 FF70 24   02       3        BCC   TERM1     GO IF NOT
00876 FF72 A7   84       4        STA   ,X        STORE CONTROL CODE
00877 FF74 39            5 TERM1  RTS

00879 FF75 E6   84       4 TERMRD LDB   ,X        GET STATUS
00880 FF77 54            2        LSRB             INPUT BIT TO C
00881 FF78 24   FB       3        BCC   TERMRD    LOOP IF NO INPUT
00882 FF7A A6   01       5        LDA   1,X       GET CHARACTER
00883 FF7C 39            5        RTS

00885 FF7D E6   84       4 TERMWT LDB   ,X        GET STATUS
00886 FF7F C5   02       2        BITB  #2        READY FOR OUTPUT?
00887 FF81 27   FA       3        BEQ   TERMWT    LOOP IF NOT
00888 FF83 A7   01       5        STA   1,X       OUTPUT CHARACTER
00889 FF85 39            5        RTS

00891 FF86 A6   84       4 TERMST LDA   ,X        GET STATUS
00892 FF88 84   03       2        ANDA  #3        MASK OFF READY BITS
00893 FF8A 39            5        RTE

00895                               ********************************************
00896                               * PSYMON COMMAND TABLE                     *
00897                               ********************************************
00898 FF8B      01       CMDTBL FCB   1           ITEM LENGTH
00899 FF8C      40              FCB   '@         MEMORY EXAMINE CHANGE
00900 FF8D      FE38            FDB   MEMBC
00901 FF8F      47              FCB   'G         GOTO ADDRESS
00902 FF90      FE71            FDB   GO
00903 FF92      4C              FCB   'L         PROGRAM LOAD
00904 FF93      FCB1            FDB   TLOAD
00905 FF95      53              FCB   'S         PROGRAM SAVE
00906 FF96      FDAC            FDB   TSAVE
00907 FF98      52              FCB   'R         REGISTER EXAMINE/CHANGE
00908 FF99      FE9A            FDB   REGEC
00909 FF9B      42              FCB   'B         SET/PRINT BREAKPOINTS
00910 FF9C      FF06            FDB   SETBK
00911 FF9E      55              FCB   'U         UNSET BREAKPOINTS
00912 FF9F      FF40            FDB   UNSBK
00913 FFA1      5A              FCB   'Z         CALL DOS
00914 FFA2      C000            FDB   $C000
00915 FFA4      FF              FCB   $FF        END  ENTIWEL

00917                               ********************************************
00918                               * RAM INITIALIZATION DATA                  *
00919                               ********************************************
00920 FFA5      43       RAMINT FCC   'CN' CONSOLE DCB ID
      FFA6      4E
00921 FFA7      FF62            FDB   TERMDR    CONSOLE DRIVER
00922 FFA9      8004            FDB   TERMNL    CONSOLE I/O ADDRESS
00923 FFAB      0000            FDB   0         ERROR STAT B, EXT
00924 FFAD      A05E            FDB   CONDCB    DCB CHAIN POINTER
00925 FFAF      A05E            FDB   CONDCB    DCB POINTERS
00926 FFB1      A05E            FDB   CONDCB
00927 FFB3      A05E            FDB   CONDCB
00928 FFB5      A05E            FDB   CONDCB
00929 FFB7      FE8E            FDB   TRAP      INTERRUPT VECTORS
00930 FFB9      FE8E            FDB   TRAP
00931 FFBB      FE82            FDB   INTRET
00932 FFBD      FE8E            FDB   TRAP
00933 FFBF      FE83            FDB   BRKPNT
00934 FFC1      FE8E            FDB   TRAP
00935 FFC3      FC32            FDB   MONENT

00937                               ********************************************
00938                               * INTERRUPT HANDLERS                       *
00939                               ********************************************
00940 FFC5 6E   9F A072  9 SWI3   JMP   [SWI3V]  SOFTWARE INTERRUPT 3
00941 FFC9 6E   9F A074  9 SWI2   JMP   [SWI2V]  SOFTWARE INTERRUPT 2
00942 FFCD 6E   9F A076  9 FIRQ   JMP   [FIRQV]  FAST INTERRUPT REQUEST
00943 FFD1 6E   9F A078  9 IRQ    JMP   [IRQV]   INTERRUPT REQUEST
00944 FFD5 6E   9F A07A  9 SWI    JMP   [SWIV]   SOFTWARE INTERRUPT
00945 FFD9 6E   9F A07C  9 NMI    JMP   [NMIV]   NON-MASKABLE INTERRU T

00947                               ********************************************
00948                               * SOFTWARE VECTORS                         *
00949                               ********************************************
00950 FFE0                  ORG   $FFE0
00951 FFE0      FD73        FDB   DSPSBY    DISPLAY SINGLE BYTE
00952 FFE2      FD6A        FDB   DSPDBY    DISPLAY DOUBLE BYTE
00953 FFE4      FD12        FDB   GETHEX    GET HEX NUMBER FROM CONSOLE
00954 FFE6      FD97        FDB   PSTRNG    PRINT STRING TO CONSOLE
00955 FFE8      FD44        FDB   INCHR     INPUT CHARACTER
00956 FFEA      FD58        FDB   OUTCHR    OUTPUT CHARACTER
00957 FFEC      FD63        FDB   REQIO     PERFORM I/O REQUEST
00958 FFEE      FC32        FDB   MONENT    MONITOR RE-ENTRY
```

PERCOM DATA CO. COPYRIGHT (c) 1979
M6800-M6809 CROSS-ASSEMBLER 3.0
PAGE 020 PSYMON PERCOM SYSTEM MONITOR FOR THE 6809

```
00960                               ********************************************
00961                               * HARDWARE VECTORS                         *
00962                               ********************************************
00963 FFFD      FC00        FDB   INIT      RESERVED BY MOTOROLA
00964 FFF2      FFC5        FDB   SWI3      SOFTWARE INTERRUPT 3
00965 FFF4      FFC9        FDB   SWI2      SOFTWARE INTERRUPT 2
00966 FFF6      FFCD        FDB   FIRQ      FAST INTERRUPT REQUEST
00967 FFF8      FFD1        FDB   IRQ       INTERRUPT REQUEST
00968 FFFA      FFD5        FDB   SWI       SOFTWARE INTERRUPT
00969 FFFC      FF09        FDB   NMI       NON-MASKABLE INTERRUPT
00970 FFFE      FC00        FDB   INIT      RESTART

00972           0000        END

TOTAL ERRORS   00000
TOTAL WARNINGS 00000

**** SYMBOL TABLE
BEGADD A038  BPTABL A034  BPTEND A052  BRKPNT FE83  CEDCB  A06C  CIDCB  A06A
CKSUM  A057  CMDTBL FF8B  CNTLFN 0008  CODCB  A06E  COMAND A056  COMP1  FC76
COMP2  FC7F  COMPAR FC74  CONDCB A05E  CONFIG 0011  CR     000D  CRLF   FDA2
DCBAPP 000A  DCBCHN A068  DCBDID 0002  DCBDVR 0004  DCBERR 0008  DCBEXT 0009
DCBIOA 0006  DCBLNK 0000  DSPBK  FF23  DSPBK1 FF28  DSPBK2 FF2F  DSPDBY FD6A
DSPID  FEFF  DSPSBY FD73  ENDADD A05A  ERROR  FCB2  FIRQ   FFCD  FIRQV  A076
GETCMD FC3A  GETHEX FD12  GETHX  FDDE  GETHX1 FD16  GETHX2 FD2B  GO     FE71
GO1    FE7C  INBYTE FCF8  INCHR  FD44  INHEX  F 2D  INHEX1 FD38  INHEX2 FD3F
INHEX3 FD41  INIT  FC00  INIT1 FC06  INIT2 FC11  INITBP FF34  INTRET FE82
IRQ    FFD1  IRQV  A078  LF   000A  LOAD   FCB7  LOAD1  FCB9  LOAD2  FC9C
LOAD3  FCDE  LOAD4  FCEB  LOAD5  FCF0  LOADX  FCF7  LOOK1  FC63  LOOKUP FC54
MEMEC  FE38  MEMEC1 FE3F  MEMEC2 FE60  MEMEC3 FE70  MEMPTR A052  MONENT FC32
NEXTBP FF39  NMI   FFD9  NMIV  A07C  OUTBYT FE22  OUTC   FE35  OUTCH  FF02
OUTCH1 FD5D  OUTCHR FD58  OUTCHX FD79  OUTDIG FD8B  OUTHEX FD7D  OUTSBY FE2C
OUTSP  FD75  PROMPT FC4E  PSTRNG D97   RAM    A000  RAMINT FFA5  RORGPF 0013
RDRON  0051  RDUMP  FEEA  RDUMP1 FEF4  RDUMP2 FEFA  READFN 0001  REGA   A029
REGB   A02A  REGC   A028  REGD   A02B  REGDMP FED3  REGEC  FE9A  REGEC1 FEA4
REGEC2 FEAF  REGEC3 FEC4  REGECX FEC6  REGIDB FEC7  REGP   A032  REGU   A03C
REGX   A02C  REGY   A02E  REMBK  FF43  REMBK1 FF45  REMBK2 FF4E  REMBK3 FF5C
REQIO  FD63  RESET  0003  RESTRT A07E  RGDMP1 FED7  RON1   FC00  ROM2   FB0C
SAVE   FDE1  SAVE1  FDE4  SAVE2  FDF9  SAVE3  FDFB  SAVE4  FE0C  SEARCH FC6A
SERCN1 FC68  SERCHX FC73  SETBK  FF06  SETBK1 FF0D  SETBK2 FF17  SP     0024
STACK  A020  STATFN 0004  STKPTR A05C  SW1    FFD5  SWI2   FFC9  SWI2V  A07A
SWI3   FFC5  SWI3V  A072  SWIV  A07A  TERM1  FF74  TERMDR FF62  TERMNL 8004
TERMRD FF75  TERMST FF86  TERMWT FF7D  TLOAD  FCB1  TPDCB  A070  TRAP   FE8E
TSAVE  FDAC  TSAVE1 FDBB  TSAVE2 FDBE  TSAVE3 FDCE  TSAVE4 FDD8  UNSBK  FF40
UNSET  FF55  UNSET1 FF61  USRTBL A054  WRITFN 0002
```

offering a translator program which converts files on soft-sectored minidiskettes for use with Percom LFD-400[tm] hard-sectored mini-disk drive systems.

The program, called SOFTRAN;[tm] is available in versions to convert files operating under mini FLEX;[†] FLEX 2.0[†] and Smoke Signal Broadcasting Company's DOS.

The significance of SOFTRAN;[tm] according to Mauch, is that it makes the Percom LFD-400[tm] a universal mini-disk storage system.

Mauch said now minidiskette programs from all of the principal 6800 software suppliers -- Technical Systems Consultants (TSC), Computerware, Ed Smith's Software Works and of course Percom -- may be used with the LFD-400.[tm]

SOFTRAN[tm] copies soft-sectored minidiskettes track-for-track onto hard-sectored minidiskettes. If the minidiskette includes a FLEX[†] or SSB DOS, the DOS is modified to function with the Percom LFD-400.[tm]

Translation of mini FLEX[†] and SSB DOS minidiskettes results in more than 10% additional storage space becoming available than required for the soft-sectored version.

SOFTRAN[tm] is supplied on a minidiskette along with utilities for $24.95. A users manual is included. The user must indicate whether SOFTRAN[tm] is to be used for mini FLEX;[†] FLEX 2.0[†] or SSB DOS.

Orders for SOFTRAN[tm] may be placed by calling Percom toll-free on 1-800-527-1592, and may be paid by check, money order, COD or charged to a Visa or Master Charge account. Texas residents must add 5% sales tax.

Dealer inquiries are invited.

tm tradeamark of Percom Data Company, Inc.

† trademark of Technical Systems Consultants, Inc.

Technical Systems Consultants, inc.

new product announcement

Box 2574 W. Lafayette, IN 47906

6809 Systems Software

Technical Systems Consultants has announced the first in a line of 6809 systems software. Initially for the SWTPc 6809 system, this software includes a 6809 version of the popular FLEX™ disk operating system, a text editor, a resident assembler, a fast BASIC interpreter, and an assembly language debug package.

The 6809 version of FLEX is almost identical to the 6800 version except for its location of C000 to DFFF hex. This implies that most software written for 6800 FLEX can be reassembled for 6809 by simply changing any equates into FLEX to the proper addresses. FLEX features include dynamic file allocation, random and sequential files, printer spooling, batch job type program entry, automatic space compression, user startup facility, and English error messages.

The text editor and assembler are included with the FLEX package but may be purchased separately. The editor is an improved version of the widespread TSC 6800 Text Editing System. It is line and content oriented with character string, local, and global instructions. The resident 6809 assembler supports macros and conditional assembly. It accepts 6809, 6800, and 6801 mnemonics so that existing software can be immediately reassembled to produce 6809 object code.

The BASIC is a very fast interpreter with 9-digit precision in its binary floating point math package. Features include random access files via record I/O and virtual arrays, unlimited string length, if/then/else construct, TRACE, ON ERROR GOTO, two-dimensional arrays, and a renumber facility. Also supported is a COMPILE command which places an unreadable, compressed form of the source on disk which can only be executed by a CHAIN command. This permits the distribution of proprietary BASIC programs.

The debug package is a powerful tool for assembly language program debugging capable of simulating all functions of the 6809 CPU including interrupts and I/O operations. Multiple breakpoints may be user-defined and may be conditional on several criteria. Tracing is possible on to single or multiple stepping. A "trace back" feature allows the debug of the previous 255 executed instructions. Memory protection may be enabled for multiple blocks of the address space. General features include a mini-assembler, disassembler, memory examine and change, hex calculator, and a machine status command.

Except for FLEX, all code is fully reentrant and position independent. FLEX is available for the SWTPc disk systems on a 5 or 8 inch disk. The other software is available on a standard 5 or 8 inch FLEX disk which may be used on any soft-sectored 6800 FLEX disk system. Cassette versions (with restricted features) are available for all but FLEX. Contact Technical Systems Consultants, Inc. P.O. Box 2574, West Lafayette, IN 47906, (317) 463-2502 for further details.

Ed Smith's **SOFTWARE WORKS** 330 Camino De Los Colinas, Redondo Beach, California 90277 (213) 373-3350

September 26, 1979

Mr. Don Williams
'68 Micro Journal
3018 Hamill Road
Hixson, TN 37343

Dear Don:

I want to thank you and Mr. Robert Boyd for the very nice comments on our Disassembler/Source Generator program in the September issue and to inform your readers that it is still available as M68SG on cassette ($25) or mini floppy ($30.95). It is available on SSB, Mini-Flex (TM) or Flex 2.0 format directly from us or in Percom format from Percom. It is also available directly from Smoke Signal, as p ogram SG.1.

The Relocatable Disassembler and Segmented Source Text Generator (X68RS) mentioned in Mr. Boyd's article is an improved (in capability) version for use in conjunction with any of our Relocatable Assembler programs. It is furnished in relocatable object code and requires the linking loader (part of the assembler package) for use. Its price is $35 on cassette or $40.95 on mini floppy and includes an annoted assembler listing of the program.

The cassette versions are delivered with the disk supproutine calls linked to Smoke Signal DOS68 entry points. Fortunately, there are enough similarities between Flex and SSB that to change to a different DOS it is only necessary to change the SSB DOS address in the disk subroutines to equivalent address in either Flex 2.0 or mini-Flex. Only 5 different addresses are used. These are:

SSB NAME	FLEX NAME	SSB	MINI FLEX	FLEX 2.0
DFA	FMS	7786	7806	B406
ΔTYPE	RPTERR	72A9	713C	AD3F
ΔFLSPC	GETFILE	7291	7127	AD2D
ΔLINEI	INBUFF	7285	7115	AD1B
PCB	PCB	7080	7740	A940

The only other change required is in the function code for "Open to Write" from 01 to 02 and for "Close File" from 0) to 04 and remove (NOP) the increment of the function code (INX 0,X) after the SSB "Open to Write" is performed.

Yours truly,

Ed Smith
General Manager

ES/js

Solar Computer Systems Corporation announces a complete series of software programs of special interest to radio and television stations designed to run on Smoke Signal Broadcasting's Chieftan Systems.

Available programs include audience measurement, attitude research, music research, lifestyle surveys, ARBITRON analyses and more.

SOLAR COMPUTER SYSTEMS CORPORATION
2360 - 43rd Avenue East
Suite #308
Seattle, Washington 98112

(206) 322-2241

Whitethorn,
3 Lemon Road,
North Balwyn, Vic. 3104.
Australia.
Telephone (03)857-7128.
29th September, 1979.

LETTERS

Dear Mr. Williams,

Referring to the problem of the "CLR Quirk" in the September issue.

The Motorola "Pink Book" (M6800 Micro-omputer System Design Data - Motorola Inc. 1976.), page 36, shows that the CLR is one of a group of instructions which also include ASL, ASR, COM, DEC, INC, LSR, NEG, ROL, ROR and TST. These all follow the same sequence for both the extended mode (6 steps) and indexed mode (7 steps). For the extended mode the sequence is:

1. Fetch Op. Code.
2. Fetch upper byte of memory address.
3. Fetch lower byte of memory address.
4. Read current contents of memory.
5. Internal operation, depending on instruction.
6. Write modified data back into memory, except for TST (dummy cycle).

Incidently, it looks as if you can expect the same problem with the 6809, although I haven't tried it yet. CLR is part of a similar group of instructions, except that TST is removed and LSL (Logical Shift Left) added, that go through a sequence involving a READ before the WRITE.

The CLR instruction appears to be the only one to cause this problem, as we need to read the memory contents so we can increment it or shift it or test it, although we might just forget about this step when writing the program.

Two more quirks you might like to think about. Don't put a PIA address at the start of a block of memory, you might get careless and put an RTS in the byte before it. And don't use VMA,P2 as an Enable for a PIA. Both of these can cause you to miss an interrupt. And don't forget that if you get an NMI while servicing a SWI the 6800 will jump to the IRQ routine. I'd tell you more, but I'm running out of space. For the answers have a look at Appendix A of the M6800 Microprocessor Applications Manual - Motorola Inc. 1975, pps A1, A7 & A10.

Keep up the good work with the magazine. I've only seen issues nos. 4 & 7 so far, it appears there is only one organization bringing it into Australia at present, and these are the only copies he has had. I hope you will keep on with material for the 6809 and 68000 in due course, and that we see some more copies here soon.

Yours sincerely,

Alan M. Fowler

R. Lynn Smith
2405 Joseph Drive
Clinton, MD. 20735

9/23/79

Dear Mr. Williams,

Many thanks to you and Dan Johnson for 'SCOPE' in the Sept. issue. There is only one problem, the patch requires some modification to work with systems using a Smartbug monitor. At the risk of over-filling the Journal with BS (backspace) articles I've enclosed a listing of Dan's program that I have rewritten to work with Smartbug on DOS68 versions .31 and .42.

The patch has been written to append directly to the DOS in place of the 'SBF' overlay used with the Smoke Signal DOS and Smartbug.

In my system a control H or $08 is used for a backspace character. This character may be changed at addresses $74F6 and $74FE to $0F or whatever other code your terminal may require. The DOS BREAK codes, as supplied from Smoke, are control X and control U. I have replaced the control U with a 'grave accent' ($60) code simply for the convenience of a single key break. This may be changed by replacing the $60 at address $7488.

Sincerely,

R. Lynn Smith

September 29, 1979

'68 Micro Journal
3018 Hamill Road
Nixson, Tennessee 37343

Gentlemen

Since my planned 6809 system is not up and running with a word processor program, this is just an old fashioned typewriter letter.

As I do not yet have a 6809 chip I cannot check this out, I hope someone can. In indexed addressing the auto increment codes are listed as only being useable in the double modes when using indirect addressing In other words this is "illegal";

LDA [0,X+]

The index code to try for this would be 10000000 or 80 hex for the post byte. There is a slight error in the L1PE program in the September issue. assembly line 00162 was left out. It should appear like this:

00162 0103 C6 50 2 UP LDB #80 *LINE COUNT

I hope you have more articles featuring the otherbus (EXORCISER) besides the SS-50 (since I use Motorola equipment).

Good work on the journal.

Darin Adler
2765 Marl Oak Drive
Highland Park, IL 60035

Gilbert A. Davis
3042 Sal Pro Rd. #3
Silver Spring, Md. 20906
Sept. 29, 1979

'68' Micro Journal
3018 Hamill Rd.
P.O. Box 849
Nixon, Tenn. 37343

Dear Sirs:

The article in your Aug. '79 issue on pg. 13 titled "Patch disk save & load T30 tape basic" has an error in it, devastating to M6800 systems. I used the basic idea to patch in my own tape on & off routine, and there is where the error reared its head. The bug lies on pg. 15 at address 8C004. The correct code should be:

C004 EE 01 LDX $1,X Put return addy. in X

What happens is the SP contains the next available memory address in the stack. What we want is the last two bytes placed on the stack by the BSR JSR instruction, which is why you must load X one byte off of X. This will get the correct data off the stack. I am also enclosing my tape on & off routine in case it may be useful to others.

Thank you for your consideration.

Yours truly,

Gilbert A. Davis
301-440-6858

```
0118  7: 5FCB  TAPCOM  JMP    ON

011B  7E 5FF6  Tapeoff JMP    OFF

5FCB  36       ON      PSHA
5FCC  FF 5FFB          STX    @5FFB
5FCF  30               TSX
5FD0  EE 01            LDX    $1,X
5FD2  8C 071C          CPX    #$071C
5FD5  27 08            BEQ    PUNCH   Called by SAVE
5FD7  8C 0785          CPX    #$0785
5FDA  27 13            BEQ    READ    Called by LOAD
5FDC  7E 0903          JMP    $0903   Basic error routine
5FDF  86 12    PUNCH   LDAA   #$12    DC2 for punch on
5FE1  BD 010F          JSR    OUTCH
5FE4  CE FFFF          LDX    #$FFFF
5FE7  09       DEL     DEX
5FE8  26 FD            BNE    DEL     Delay for motor
5FEA  FE 5FFB  EXIT    LDX    @5FFB   Restore X
5FED  32               PULA
5FEE  39               RTS
5FEF  86 11    READ    LDAA   #$11    DC1 reader on
5FF1  BD 010F          JSR    OUTCH
5FF4  20 F4            BRA    EXIT
5FF6  36       TAPOFF  PSHA
5FF7  86 13            LDAA   #$13    DC3 reader off
5FF9  BD 010F          JSR    OUTCH
5FFC  32               PULA
5FFD  39               RTS
```

My Hardware decodes either a DC3 or DC4 code to turn off the relay to my cassette. That is why I use only one code in the tape off routine. The first part of the tape on routine could be duplicated to support a seperate punch & reader relay. My top of memory is at 05FFF, so that is where my routine lies. It can be placed anywhere.

Mr. Anthony Miesz
31 Winnett Street
Hamden, CT 06517

September 28, 1979

Editor
68 Micro Journal
3018 Hamill Road
P.O. Box 849
Nixson, TN 37343

Dear Sir:

I just want to let you know how very much I like your publication. I think your plan of publishi g the back issues in a single volume is excellent, and is clearly the only way to go. Thanks for publishing the review of the SWTPC 6809 board. I would get one immediately except that I have too much software for the 6800 which I would have to convert to 6809 format. Would it be possible to switch select between the 6800 and the 6809 board, or are the mother board changes too extensive to permit this to be done? (I have the SWTPC CT-82, the SWTPC 6800/2, and the MP-68, with 40K RAM.)

DOUBLE DENSITY

The most reliable, cost effective disk system ever designed for the SS-50 bus is now available. The Southwest Technical Products Corp. DMF-2 disk system provides 2.5 M/bytes of usable (formatted) on-line storage. It offers the lowest cost per byte available on floppy disks at this time.

The DMF-2 features "Oume" DATATRAK 8 double headed eight-inch drives. We consider these to be the best drives we have ever tested. The 17½" x 5" x 21½" cabinet is made from 1/8 inch thick aluminum and finished with a super tough textured epoxy. The power supply has 115/230 volt capability and will operate from either 50 or 60 Hz. mains.

The controller is a direct memory access type circuit, using the 6844 DMA controller and a 1791 double density disk controller. This type circuit has a much higher data transfer rate than simple sector buffer type circuits and it also imposes far less overhead on the processor. The critical phase lock and data separator circuits use 1% components and time proven circuits to insure long term reliability. We find no statistical difference in the error rate of this controller and our single density controller.

The DMF-2 is supplied with the FLEX®-09 operating system. You can format and record in either single or double density. FLEX® is the world standard disk operating system for the MC6809 and is available for almost all 6809 family hardware, whatever the source.

The DMF-2 system includes the cabinet, power supply, controller, connecting cable, diskette with FLEX®-09, two drives and instruction manual. Shipping weight is 53 pounds.

DMF-2 Dual Double Density Disk Driver—assembled and tested $2,495.00
DMF-2 Controller board retrofit for converting DMF-1 single density systems . . . . . $ 395.00

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Print with Quality and Speed

The Southwest Technical Products fast quality printer system is based on the "Qume" Sprint 3/45 daisy-wheel printer. For word processing applications, where quality and speed are both necessary , this printer is the answer. Over a hundred styles of printer wheels are available, including proportional space and foreign type fonts.

The SP-3 is supplied with the following features: out of paper detect, out of ribbon detect, top of forms elect, bottom feed slot, cover interlock, operator lights, paper handling system and switching power supply. Optional forms tractors are available for applications where these are desired. The SP-3 printer is supplied with a twelve-line interface and connecting cable for use with all Southwest Technical Products computers.

- Average text print speed of 45 characters/second
- Prints full characters of electric typewriter quality
- Uses variable intensity ballistic hammer which automatically adjusts to correct one of six strike intensities according to character size
- Accepts single sheets and continuous forms, with or without sprocket holes
- Prints on forms up to 15 inches wide
- 96 character positions on "daisy" printwheel
- Wide variety of standard font styles available in 10 and 12 pitch and proportional spacing
- Prints 132 columns at 10 characters/inch
- Prints 158 columns at 12 characters/inch
- Prints proportional spacing in increments of 1/120 inch, left or right
- Features electronic tabbing and carriage return up to 13.1 inches at 320 ms maximum
- Vertical spacing in increments of 1/48 inch, up or down
- Vertical slew rate of 5 inches per second
- Plotting resolution of 5760 points per square inch
- Features pressure platen; pin feed platens optional
- Easy to handle ribbon cartridge with multi-strike carbon, single strike carbon, or fabric ribbon available in black and colors
- Printwheel is easily operator changeable
- Operator controls include horizontal forms positioning, vertical forms positioning, forms thickness and ribbon advance

SP-3	Daisywheel Printer—with listed features, interface and power supply	$2,995.00
SP-5	Serial Daisywheel Printer—with above features and power supply (less serial interface)	$3,195.00
80026-01	Optional Forms Tractor	$ 190.00

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Although I built everything except the CT-82 myself, I am basically a beginner. (I use the computer mostly for text processing--I teach foreign languages, and have no previous computer experience.) Almost all of my own programming efforts have been limited to BASIC. Although I really like the speed of the new T&C BASIC running under FLEX 2, there are times when I would prefer the increased arithmetic accuracy of the SWTPC disk basic, which is much slower but has BCD arithmetic. Is there anyone out there who has converted the SWTPC Disk BASIC (Version 3) to run under FLEX 2? T&C doesn't provide conversion instructions, and since SWTPC doesn't provide the source code, such a conversion is a bigger project than most of us would care to take on.

I've only recently started to become interested in assembly language programming, in which I am even more of a neophyte than in BASIC. I am enclosing my first program (actually, it's too small and too easy to call it that) which is designed to send the appropriate control character sequence to the CT-82 terminal to configure it on startup for my major use, text processing. Perhaps you might want to print it as an example of a most elementary assembler program. (Since the program is so simple, I wouldn't expect your usual extension of subscription reward.) Naturally, any user could substitute his own choice of control character sequence for any of mine, or simply add some additional ones as necessary.

I am looking forward to your coming software reviews, particularly of the Pascal advertised by Lucidata. Is it any good? I really don't know if I want to learn Pascal yet, but if it is really as powerful as SWTP is always saying, perhaps one should look into it. My little terminal configuration routine is attached.

Best of luck to you.

Sincerely yours,

Anthony [illegible]

```
                            NAM     CT-82.CMD
                    *  BY TONY NIESS
                    * THIS PROGRAM WILL SEND THE APPROPRIATE CONTROL
                    * CHARACTER SEQUENCE TO THE SWTPC CT-82 CRT TO
                    * CONFIGURE IT AS DESIRED.
                    *
0000                EQUATES
                    *
AD03                FLEX    EQU     $AD03
AD1E                PSTRNG  EQU     $AD1E
                    *
0100                        ORG     $0100
0100 CE 01 1A       CFIG    LDX     #NOBLNK     SET NON-BLINKING CURSOR
0103 BD AD 1E               JSR     PSTRNG
0106 CE 01 1D               LDX     #NRMSHF     DISABLE SHIFT INVERSION
0109 BD AD 1E               JSR     PSTRNG
010C CE 01 20               LDX     #LWINTN     SET LOW INTENSITY
010F BD AD 1E               JSR     PSTRNG
0112 CE 01 23               LDX     #NCHRPR     IGNORE CHARACTER PROTECTION
0115 BD AD 1E               JSR     PSTRNG
0118 20 0C                  BRA     RETRN
011A 1E             NOBLNK  FCB     $1E,$13
011B 13
011C 04                     FCB     $04
011D 1C             NRMSHF  FCB     $1C,$10
011E 10
011F 04                     FCB     $04
0120 1C             LWINTN  FCB     $1C,$05
0121 05
0122 04                     FCB     $04
0123 1E             NCHRPR  FCB     $1E,$07
0124 07
0125 04                     FCB     $04
0126 7E AD 03       RETRN   JMP     FLEX
                            END     CFIG

NO ERROR(S) DETECTED

   SYMBOL TABLE:

CFIG   0100   EQUATR 0000   FLEX   AD03   LWINTN 0120   NCHRPR 0123
NOBLNK 011A   NRMSHF 0110   PSTRNG AD1E   RETRN  0126
```

R. Desmarais

After upgrading my SWTPc 6800 with their 6809 board and checking out the basic functionality of the T&C software, I decided to time the T&C 6809 BASIC with the seven benchmark programs used in Issue #10 of KILOBAUD. In order to obtain accurate times, I extended the program I have to prompt with the time of day (see June issue of 68 Micro Journal) to clock at 1/60 sec. instead of at 1 sec. By modifying the benchmark programs to display the time instead of the "START" and "END" messages, I was able to obtain extremely accurate times. The table below summarizes the timings I obtained along with those given in KILOBAUD for the SWTPc 8K BASIC (1.0).

Benchmark #	1	2	3	4	5	6	7
6800 BASIC	14.9	24.7	96.1	105.3	109.8	174.1	204.5
6809 BASIC	.267	3.5	10.99	11.3	11.93	17.59	26.9
6809 BASIC (compiled)	.233	3.417	10.9	11.2	11.817	17.25	26.533

The 6809 times above are the average of three consecutive runs of the programs.

It should be noted that the time difference between two successive requests for a time display via the USR function is .183 sec.

The actual program used for Benchmark Program 1 is shown below:

```
100 E=USR(0)
400 FOR K=1 TO 1000
500 NEXT K
700 E=USR(0)
800 END
```

R. Desmarais
12 Richard Rd.
Medway MA 02053

weldy moffatt

1111 college ave
regina sask canada
S4P 1A8

1979 sep 20
'68' MICRO JOURNAL

I submit a program in SWTPC BASIC 3.0 or your consideration. As secretary of a service club, I am called upon to run the business end of a football pool. Following each game there was the job of finding out who the holders of the winning scores were and for a typical game it took about 2 hours to search through the ticket stubs to find the scores that matched with the game scores.

The printer of the pool tickets had printed 1600 combinations with the series decrementing by one point for the home team and incrementing after each fourtieth ticket for the visiting team. There are 19 games in the schedule and the pool tickets are sold as a booklet of 19 tickets in a numbered book and varying base starting points for each game. In past years the tickets had been selected at random to find one that had a winning score then the others were found by branching out.

It was obvious to me that the printer was using a mathematical formula to print the tickets and that by examining booklet 1600 and treating it as 0 in a formula all of the booklet numbers for the valid winning scores could be calculated. A simple BASIC program proved this and this program is the one that has been used for the 1979 games.

This program should be easily converted for any pool with tickets numbered by formula. The line 720 the home score is "H" and the opposition is "O" while line 230 generates the booklet number. Line 1145 provides an easy method of aborting the program by entering a score in excess of 500. Lines 7000 and 8000 convert invalid booklet numbers to their equivalent valid number.

This program assumes use of a video terminal and a keyboard for data entry and a printer for creating a file copy of the output. When used with a video terminal alone, all references to alternate port should be taken out and a suitable stop inserted after printing of data on terminal. I like to use INPUT A$ then the program will continue after a carriage return.

```
                                Weldy

DATE OF WEEK  1
WEEK NUMBER   1
FIRST BOOK    1171
SCORE 1 0              BOOK #1170
DATE OF WEEK  19/9 SEP 16
WEEK NUMBER   18
FIRST BOOK    1268
SCORE 0 0              BOOK #1268
SCORE 0 1              BOOK #1308
SCORE 0 2              BOOK #1348
SCORE 0 8              BOOK #1588
SCORE 0 9              BOOK #20
SCORE 0 15             BOOK #268
SCORE 0 16             BOOK #308
SCORE 0 22             BOOK #548
SCORE 0 23             BOOK #588
SCORE 6 23             BOOK #574
SCORE 7 23             BOOK #573

0010 REM *   WINNER BAS
0020 REM *   DISK - 10
0030 REM * WELDY MOFFATT 7/SEP/16
0040 REM * BACKUP DISK-24
0050 PRINT CHR$(16);CHR$(22)
0060 LINE= 140
0070 REM ********************************
*****
0101 REM THIS PROGRAM USED TO FIND WINNI
NG
0102 REM BOOK NUMBERS IN GROUP OF 1600 B
OOKS
0103 REM FOR 19 GAMES   SCORES RANGE FRO
M
0104 REM 40 - 40 TO 0 - 0
0105 REM PROGRAM IS SELF PROMPTING AND
0106 REM EXPECTS PRINTER ON PORT 2
0107 REM AND VIDEO ON PORT 1
0108 REM TO EXIT PROGRAM USE SCORE 999
0109 REM FOR FIRST ITEM
0110 REM ********************************
*****
0190 INPUT "DATE OF WEEK";Y$
0200 INPUT "WEEK NUMBER";G
0210 PRINT "ENTER SCORE FOR BOOK 1600 (
0.0)"
0220 INPUT A,O
0224 LET S=(1600-A)-(G*40)
0235 IF S<1 GOSUB 7000
0240 PRINT "CHECK TO MAKE SURE BOOK ";S
0250 PRINT "HOME SCORE ----------"
0260 PRINT "           |     |  "
0270 PRINT "           |     |  "
0280 PRINT "           |     |  "
0290 PRINT "           |     |  "
0300 PRINT "           ------- "
0310 PRINT "DOES THE BOOK MATCH THE PRI
NT"
E$="
0320 INPUT "(Y OR N)";X$
0330 IF X$="N" GOTO 30
0500 REM *** DUMP ON PRINTER ***
0510 PORT= 2
0520 PRINT "DATE OF WEEK  ";Y$
0540 PRINT "WEEK NUMBER   ";G
0550 PRINT "FIRST BOOK    ";S
0560 PORT= 1
1000 REM *** GET WEEK SCORES ***
1110 PRINT CHR$(16);CHR$(22)
1140 INPUT "1ST SCORE",K
1145 IF K>500 THEN END
1150 INPUT "2ND SCORE",O
1170 LET X=S-(K*1)+(O*40)
1180 IF X>1600 GOSUB 8000
1200 REM *** DUMP ON PRINTER ***
1205 PORT= 2
1210 PRINT "SCORE ";K,O,
1220 PRINT USING"BOOK #";X
1300 GOTO 590
7000 LET S=S+1600
7010 RETURN
8000 LET X=X-1600
8010 RETURN
9900 PORT= 2
9990 PRINT : PRINT PRINT
9995 PRINT : PRINT PRINT
9998 PORT= 1
9999 END
```

111 Division St., #19
King City, California
93930
20 July 1979

Dear Sirs:

Again, let me say I am grateful for the 68 Micro Journal. If it keeps on at the present rate I will be cancelling most of the other micro mags I subscribe as you seem to meet my needs and I don't have to scan through a bunch of stuff to find material and items for my machine.

I am including a printout of a small program I wrote recently for making our Christmas card labels on my printer. It is for the LFD 200 Disk system by Percom and is written in Percom's Superbasic. It creates a file on disk

of the names and addresses and will search for a specific name, display the address and ask if the user wants to make any corrections. In this manner the list can be updated as needed. It also contains provisions for the first 20 addresses to be used as a separate list, for more regularly used addresses. This is by no means a fancy sorter for businesses. It is only what it seems to be.....a simple file of addresses that can be searched and changed as necessary and that will print out the labels themselves on standard form-feed label material. I might add that this is the program that finally sold my wife on the uses of a computer in the home. It is simply marvelous to have that nasty chore of addressing the annual cards done for you! It is simple, straightforward, and should, I would guess, run with minor changes on almost any system in BASIC. I hope you will find it useful.

Prior to sending it along, I thought I would ask if you would have use(space) for another one I rather like. I have adapted a program called "Swords and Sorcery" or the 68OC, which appeared,originally, I believe, in Creative Computing (I can and will supply a copy of the original article so appropriate credit can be given the original author.) This is not quite lengthy, however, written for the LFD 400, again, and will recall who has played before, whether they won or lost, and how often, and how far they have travelled (an internal game matter), with appropriate greetings for new or old players. It is the most popular game on my system at present even though it seems to require no skill whatsoever. It takes about 14-15K to store in one piece, but I have it written in three separate sections which are 'chained' together as needed. As it is long, I thought I would ask before sending a listing to you, but would be happy to share it via the Journal.

Thirdly, I have a simple (!) household accounts program, again in BASIC with disk files, which stores the amounts and numbers of outstanding checks, provides checkbook balancing routines, etc.

I hope these might be useful, but due to the length of the latter two I thought I would ask prior to sending them along.

Again, thank you for your journal. Keep it up. I intend sending my renewal for three years, in August (to beat the price increase in September, of course.)

Sincerely,

PAUL E. PHELPS
Chaplain (MAJ), USA

```
0010 REM THIS PROGRAM WRITTEN JUNE & JULY 1979 FOR CREATION AND PRINTING
0015 REM AND EDITING OF OUR CHRISTMAS CARD LIST ADDRESSES.  IT ALSO HAS
0020 REM PROVISION FOR PRINTING THE FIRST 19 ADDRESSES ON THE LIST FOR
0025 REM SPECIAL MAILINGS.  ADDITIONS MUST BE MADE **ONLY** IN THE
0030 REM ORIGINAL LIST OR THE 2 OR 3 BLANK LISTINGS AT THE END OF THAT
0035 REM PORTION OF THE LIST.  IF THEY ARE TO BE PART OF THAT SEPERATE
0036 REM LIST. ALL OTHER ADDITIONS OR DELETIONS WILL OCCUR AT THE END
0037 REM THE REGULAR FILE.  NAMES AND ADDRESSES MUST NOT BE LONGER THAN
0040 REM THE ORIGINAL ENTERY PLUS ITS EMPTY SPACES (DIVIDED BY 16) ON
0045 REM THAT LINE, ELSE THE FOLLOWING LINES WILL BE MESSED UP.
0050 REM WRITTEN IN SUPERBASIC+ FOR THE LFD 400 DISK DRIVE BY
0055 REM PAUL E. PHELPS 111 DIVISION ST. #19, KI 8 CITY, CALIFORNIA
0060 DEF 83030
0065 REM '0' IS PROPER ENTRY FOR HAVING NOTHING TO ENTER.
0080 OPEN #10,"XMASFL",RESTORE #10
0095 LINE= 0
0100 PRINT " WHAT DO YOU WANT TO DO TODAY:"
0110 PRINT TAB(15)"CHANGE ADDRESSES OR SPELLINGS - TYPE - CHANGE"
0120 PRINT TAB(15)"ADD NAMES TO THE LIST - TYPE - ADD"
0130 PRINT TAB(15)"PRINT THE WHOLE LIST ON LABELS - TYPE - PRINT"
0135 PRINT TAB(15)"PRINT ONLY SPECIAL LIST ON LABELS - TYPE - SPECIAL"
0140 PRINT TAB(15)"FINISH FOR THIS TIME - TYPE - FINISH"
0200 INPUT A$
0210 IF LEFT$(A$,1)="C" THEN 2000
0220 IF LEFT$(A$,1)="A" THEN 1100
0230 IF LEFT$(A$,1)="P" THEN W=200 : GOTO 3000
0240 IF LEFT$(A$,1)="S" THEN W=23 : GOTO 3000
0250 IF LEFT$(A$,1)="F" THEN CLOSE #10 : END
1020 REM THIS IS THE CHANGE SPELLING OR ADDRESS PROGRAM
1100 READ #10,C8,T200
1110 GOTO 1100
1200 INPUT "NAME:";D$
1210 IF D$="0" THEN 100
1220 INPUT "ADDRESS:";E$
1230 INPUT "CITY   :";F$
1240 INPUT "STATE AND ZIP:";G$
1250 PRINT #10,D$,C7,E$,C7,F$,C7,G$,
1260 PRINT #10,"","",""
1270 GOTO 1200
2000 REM THIS IS THE SEARCH AND DISPLAY ON VIDEO PROGRAM.
2010 REM IT ALSO ALLOWS FOR CHANGES IN NAME & ADDRESS IF NECESSARY.
2100 INPUT "EXACT NAME PLEASE:";NM$
2110 IF NM$="0" THEN 100
2120 LET X=LEN(NM$)        REM THIS ADDS SPACES TO NM$ SO IT WILL EQUAL
2130 IF X=16 THEN 2300!    REM WHAT IS STORED ON DISK
2140  FOR I= X+1 TO 16-1 : GOSUB 2190
2150  T=32-I : GOSUB 2190
2160  T=40-I : GOSUB 2190
2170  T=56-I : GOSUB 2190
2180 IF SG$(I)=" " THEN 2200
2185 RETURN
2200 FOR I=1 TO I-1
2210 LET NM$=NM$+" "
2220 NEXT I
2300 RESTORE #10
2310 READ #10,C$,A100
2320  N=SCIN : N=N+1
2330 IF C$=N$ THEN RESTORE #10,N :GOTO 2350
2340 GOTO 2310
2350 READ #10,C$,A0FF
2360 IF C$="""""" THEN 2390
2370 PRINT C$
2380 GOTO 2350
2390 PRINT #7:?
2400 INPUT "ANY CHANGES NECESSARY ";A$
2410 IF LEFT$(A$,1)="Y" THEN 2500
2420 GOTO 2000
2500 RESTORE #10,N
2510 INPUT "NAME   :";D$
2520 INPUT "ADDRESS:";E$
2530 INPUT "CITY   :";F$
2540 INPUT "STATE AND ZIP :";G$
2550 PRINT #10,D$,C7,E$,C7,F$,C7,G$,
2560 PRINT #10,"","",""
2570 GOTO 2000
3000 REM THIS IS TO PRINT LABELS FOR THE CARDS
3020 RESTORE #10
3030 PORT= 1
3040 READ #10,C$,A3F00
3050 IF C$="""""" THEN 3000
3060  D=0+1 : ?#1
3070 GOTO 3F00
3080 IF D=4 THEN D=0:?#1 : REM THIS SETS SPACES BETWEEN LABELS
3090 IF D=5 THEN D=0:? : REM ON LABELS 7/8 BY 3 1/2 INCHES.
3100 IF D=6 THEN D=0
3110  Z=Z+1 : IF Z=W THEN 100
3120 GOTO 3F00
3000 PORT= 1
3010 PRINT "WHAT'S THE END OF THE LIST.  YOU HAVE ";Z;" LABELS NOW."
3020  Z=0 : GOTO 100
4000 REM THESE STATEMENTS INDICATE A PROBLEM SOMEWHERE IN THE
4010 REM PROGRAM
4020 PRINT "SOMETHING IS WRONG WITH THE INPUT FUNCTION."STOP
4100 PRINT "THAT NAME IS NOT FOUND IN THE LIST IN EXACTLY THAT FORMAT"
4110 PRINT "TRY AGAIN, PLEASE."
4120 GOTO 2000
9000 REM A UTILITY PROGRAM TO PRINT WHAT IS STORED ON DISK TO THE
9010 REM VIDEO WITH SECTOR NUMBERS FOR CHECKING.
9020 OPEN #10,"XMASFL"
9030 READ #10,C$,94PP
9040 PRINT SCTR,C$
9050 GOTO 9 30
9400 STOP
```

ART WELLER
5201 CORNELL AVE
EL PASO, TEXAS 79924

'68' MICRO JOURNAL
3610 HAMILL RD.
P.O. BOX 849
HIXSON, TENNESSEE 37343

ATTN: Don Williams, Editor

Dear Don

First of all, thanks for listing programs in big print. I'm sure the rest of the bi-focal set appreciates it, too. I was able to load Jim Thomas' "Basic to DOS" (Aug issue) without a single typing error--a first for me! Works great. Which brings me to the purpose of this letter.

Please give Jim a "Versatile Patch of the Year" award. What new be the most useful aspect of his mod was not mentioned in the brief article. That is the potential to use it to add transient features and overlays to Basic. For example, I immediately adapted Mickey Ferguson's RENUMBER (with Cass' patch) and have a handy new Basic command called DC RENUMBER. We can have all kinds of extra Basic goodies without tying up an arm and a leg's worth of RAM.

I'm so delighted with this concept I have spelled it to TSC EDITOR (listing enclosed). As you can see, my next project will be to add a hard copy trick to EDIT. No problem finding space in the table as EDIT has lots of redundant commands.

Seeing the hexadecimal conversion program in the Sep issue reminded me of one I wrote a year or two ago that converts between any two bases up to 16. Don't ask why anybody'd want to go from base 7 to base 13(!), but it is handy to be able to go between any combination of 2, 8, 10, &16. If you decide to use it for filler some day you may not want to call it a "QUICKIE" as changing something like EAFF to binary is a good time to take a coffee break!

Yours truly
Art Weller

Oct 4, 1979

```
                        NAM    ED-DOS (EDITOR TO DOS CMD "DC")
                        OPT    NOG,PAG
                * FROM '68', AUG 79
                * BY JIM THOMAS - JULY 79
                * MODIFIED FOR USE WITH TSC 'EDITOR'
                * BY ART WELLER, OCT 1979
                *
                * THIS EDITOR COMMAND PROVIDES A MEANS OF
                * EXECUTING DOS COMMANDS DIRECTLY FROM
                * EDITOR.  EDITOR FEEDS THE COMMAND TO THE
                * DOS COMMAND BUFFER AND TURNS CONTROL OVER
                * TO DOS.  WHEN THE COMMAND IS FINISHED,
                * EDITOR RESUMES CONTROL.
                *
                * SYNTAX: DC <dos command line>
                *
                * EDITOR ADDRESS LOCATIONS
0214            APPADD  EQU    $0214     "APPEND" ENTRY IN TABLE
0492            SKIPSP  EQU    $0492     SKIP SPACES
0203            EDBAK   EQU    $0203     RE-ENTER EDITOR
1680            BEGPNT  EQU    $1680     STORE START OF DATA AREA
00BB            BUFFER  EQU    $00BB     EDITOR INPUT BUFFER
0044            BUFPT1  EQU    $0044     BUFFER LINE POINTER
                * ADDRESSES MAY CHANGE WITH
                * DIFFERENT VERSIONS OF EDITOR
                *
                * FLEX ADDRESS LOCATIONS
7000            DOSBUF  EQU    $7000     DOS LINE BUFFER
707F            DBEND   EQU    $707F     END OF DOS BUFF
7094            BUFPNT  EQU    $7094     BUFFER POINTER
7118            PSTRNG  EQU    $7118     PRINT STRING
7142            DOCMND  E      $7142     EXECUTE DOS AS SUB
                *
                * MAKE ROOM FOR THE DC ROUTINE AFTER EDITOR
1680                    ORG    BEGPNT    CHANGE START OF MEMORY
1680 1A 30              FDB    FINI      NEW START OF MEM
                *
                * INSERT NEW COMMANDS
0214                    ORG    APPADD    "APPEND" IN TABLE
0214 44                 FCC    'DC'
0216 00                 FCB    0
0217 19 DB              FDB    DC
```

```
0044  0219 24              FCC    '$'         RESERVE SPACE
0045  021A 00              FCB    0           FOR FUTURE HARD COPY
0046  021B 08 77           FDB    $0877       PRINT COMMAND
0047
0048                  * OLD CONTENTS OF BEGPNT TELL WHERE TO
0049                  * START THE PATCH
0050
0051  19DB                ORG    $19DB       NEXT RAM ADDRESS
0052  19DB                EQU    *           START OF PROGRAM
0053  19DB DE 44          LDX    BUFPT1      PICK UP COMMAND LINE
0054  19DD BD 04 92       JSR    $4IPSP      SKIP LEADING SPACES

0055  19E0 DF 44          STX    BUFPT1
0056  19E2 CE 70 00       LDX    #$0700BUF   LOAD DOS BUFFER
0057  19E5 FF 70 94       STX    BUFPNT      INIT DOS POINTER
0058                  * TRANSFER FROM EDITOR TO DOS'S COMMAND
0059                  * BUFFER AREA
0060  19E8 FF 1A 13  DC1  STX    TEMPX2
0061  19EB DE 44          LDX    BUFPT1
0062  19ED A6 00          LDA A  0,X         GET CHAR FROM BUFFER
0063  19EF 08             INX                BUMP TO  EXT CHA
0064  19F0 DF 44          STX    BUFPT1
0065  19F2 FE 1A 13       LDX    TEMPX2
0066  19F5 A7 00          STA A  0,X         MOVE TO COMMAND BUFF
0067  19F7 08             INX
0068  19F8 8C 70 7F       CPX    #BEND       END OF BUFFER?
0069  19FB 27 0D          BEQ    ERROR       EXIT WITH ERROR
0070  19FD B1 0D          CMP A  #$D         ARE WE DONE
0071  19FF 26 E7          BNE    DC1         NO KEEP MOVING
0072  1A01 BD 71 42       JSR    DOCMND      CALL DOS AS SUBROUTINE
0073  1A04 5D             TST B              ANY ERRORS?
0074  1A05 26 03          BNE    ERROR       YES, TELL OPR AND GO EDITOR
0075  1A07 7E 02 03       JMP    EDBAK       GO BACK TO EDITOR
0076  1A0A          ERROR  EQU    *           ERROR ROUTINE
0077  1A0A CE 1A 15       LDX    #ERST       POINT TO ERROR STRING
0078  1A0D BD 71 1B       JSR    PSTRNG      PRINT IT
0079  1A10 7E 02 03       JMP    $0203       EDITOR WARM START
0080  1A13          TEMPX2 RMB    2
0081  1A15 45       ERST   FCC    'ERROR IN DOS COMMAND LINE'
0082  1A2E 04             FCB    $4
0083  1A2F 0D             FCB    $D
0084  1A30          FIN1   EQU    *
0085                       END

NO ERROR(S) DETECTED
```

```
0010 REM VERSION 01--DEC 1978--A. WELLER
0020 PRINT "THIS CONVERTS BASE 2 TO 16 NUMBERS FROM ONE "
0030 PRINT "BASE TO A OTHER."
0040 PRINT "INPUT BLANK LINE TO CHANGE BASES."
0050 DIM H$(16)
0060 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G
0070 PRINT "FROM BASE: ";:INPUT B1
0080 PRINT "TO BASE: ";:INPUT B2
0090 PRINT :PRINT "DATA ";:INPUT D$
0100  L=LEN(D$):X=0:D=0
0110 FOR N=L TO 1 STEP -1
0120  C$=MID$(D$,N,1)
0130 IF C$>"F" GOTO 70
0140 FOR M=0 TO B1
0150 IF N=B1 GOTO 70
0160 READ H$
0170 IF H$=C$ GOTO 190
0180 NEXT M
0190  D=D+M*B1^(X):X=X+1:RESTORE
0200 NEXT N
0210  N=0
0220  Z=0
0230 IF D<B2 GOTO 250
0240  B=D-B2:Z=1+1:GOTO 230
0250  N=N+1
0260 FOR M=1 TO D+1
0270 READ H$
0280 NEXT M
0290  H$(N)=H$:RESTORE
0300 IF Z=0 GOTO 320
0310  D=Z:GOTO 220
0320 PRINT D$;" BASE ";B1;"= ";
0330 FOR L=N TO 1 STEP-1
0340 PRINT H$(L);
0350 NEXT L
0360 PRINT " BASE ";B2
0370 GOTO 90
0380 END
```

ACCUTEST Corp.
25 Industrial Ave.
Chelmsford, MA 01824
July 17, 1979

Mr. Don Williams
"68" Micro Journal
3018 Manill Road
PO Box 849
Nixon, TN 37343

Dear Don:

When I talked to you on the phone a few weeks ago, I was wondering whether you knew anything about Seals Electronics. I had ordered one of their 68KC8 bare 8k memory boards and not heard from them since. Well, Bruce Seals still does business, and his service still is good, and his boards are excellent. I kept calling his old phone number, and finally an operator intercepted and gave me his new number, (615)-584-0073.

Anyway, I did finally get in touch with him, and five days later I had five of his bare boards. They are $35.- each, use 64 2102's giving 8K by 8 bytes. As I happened to have quite a few surplus 2102's around, it was definitely what I was looking for. Your readers might want to know about them, since Bruce does not advertise them, and they are the only ones around. I have now assembled five of them, and I am waiting for a sixth. The boards do require a "fix" to address them at the top 32K, but it is an easy one. The solder mask is excellent - all five boards worked the first time they were plugged in. Additional components required are: Two 74LS138's, two DS8835's (Electrolabs have them at $1.25 ea), four 74367's (same as 8097, 8T97 etc.), four 7805 5V regulators, a 7404 and a 7402 an incredible number of bypass caps, and if you want to use the battery back-up feature, a couple of resistors and diodes plus the batteries. The boards come with Molex edge connectors, and for $60 you can get them with sockets wave soldered, if you trust sockets... (I don't recommend them). I've seen 2102's selling at $0.60 each and less in quantity, so for someone with e.g. a Cimix that has the power, it is a good idea...

In my last letter I talked about a patch to SSB's format V 1.2, that would allow you to format the minifloppies to 41 tracks instead of 35. The API drives that Sacku supplies can handle that, and it gives you another 100 sectors to play with. I have also found a couple of extra switches that can be used. Unfortunately, at the time I wrote the letter, I didn't have enough memory to reassemble my patch; so I missed one of the changes you need to make... This I guess is true whenever you try to copy something from a computer printout. I apologize. I should know better. Anyway, I have enclosed a complete assembled & commented source of the patch.

Sincerely,

Martin R. Furuhjelm

```
                        LIL          FORMAT PATCH
                        OPT     NOG,PAG
                *
                * PATCH THE FORMAT COMMAND TO FORMAT FOR 41 TRACKS
                * ON API - DRIVE
                * WARNING DO NOT USE ON SHUGART DRIVES.
                *
0300                    ORG     $300        CHANGE NUMBER OF TRACKS TO 41
0300 CE 12 29           LDX     #$1229
03F0                    ORG     $3F0
03F0 A8 51              FDB     $A851       LAST TRACK-SECTOR
03F2 02 E0              FDB     $02E0       SIZE
                *
                * UNLOCK EXTRA COMMANDS
                * THEY'RE HERE, BUT SSB DOES NOT SUPPORT THEM,
                AND THEY MAY NOT WORK.
                *
021F                    ORG     $21F
021F 49                 FCC     /I/         DON'T KNOW WHAT THIS DOES YET.
                *
0229                    ORG     $229
0229 56                 FCC     /V/         VERIFY DISK WITHOUT FORMAT
                *
0249                    ORG     $249
0249 43                 FCC     /C/         DON'T PROMPT "ARE YOU SURE" etc.
                * OW PROMPTS "WORKING ON DRIVE"
                * S METIMES FOLLOWED BY DRIVE NUMBER...
                *
                * CH NGE PROMPT TO FORMAT "MPI"
02E                     ORG     $02E
02E 4D                  FCC     /MPI /
                *
                        END
      NO ERROR(S) DETECTED
```

INEXPENSIVE DAN

Don Aldridge
Senior Applications Eng.
Tech. Training, Motorola
Phoenix, AZ 85008

INEXPENSIVE
DATA ACQUISITION NETWORK
FOR YOUR MICROPROCESSOR

INTRODUCTION

One of the most difficult components for the hobbyist computer system is interfacing the digital computer to the analog world of voltages or currents. The desirability of being able to interface to the analog world is becoming greater due to many of the new applications for the computer system. One of these recent applications is environmental control system for the home, workshop, or office.

Data acquisition or A/D modules for microprocessor are available from many sources; however, the cost of these modules is usually quite high and may exceed the cost of the computer system. These modules usually feature very high performance in accuracy and conversion speed and are an overkill for many of the hobbyist applications.

This article discusses an 8-channel DAN (data acquisition network) using the Motorola MC14433 CMOS A/D converter and M6800 microprocessor system. This system allows the hobbyist to add an inexpensive DAN module to his home computer. The number of channels of the DAN may be expanded or reduced by slight modification

of the hardware and software. The inputs to the DAN are high impedance inputs and capable of handling voltages of both positive and negative polarity. Maximum voltage input is 1.999 volts. The output of the system is 4 memory addresses in RAM for each channel. These channels contain the 3 1/2 digits of BCD information, the polarity of the input signal, and overrange information if the input voltage exceeds 1.999 volts.

Additional components for an existing M6800 microprocessor system would be a MC6821 PIA and the MC14433 A/D. To aid in understanding the DAN described here, an understanding of the MC14433 will be helpful.

MC14433 A/D CONVERTER

The MC14433 is a single chip 3 1/2 digit A/D converter using a modified dual ramp technique of A/D conversion. Housed in a 24 pin package, it features auto-polarity, autozero, and a high input impedance. Figure 1 shows the pin diagram of the MC14433.

The output of the MC14433 is 3 1/2 digit multiplexed BCD with the MSD containing not only the half digit but also polarity of the input, and overrange and underrange information. Figure 2 describes the decoding for the MSD. The digit selects for the multiplexed BCD have interdigit blanking to ensure correct BCD data during the time that the digit select is true.

The A/D converter is ratiometric and requires an external reference voltage. This reference voltage is 2.000 volts for the 1.999 volt range and 200 mV for a 199.9 mV full scale input. Both the unknown and reference inputs and analog ground are high impedance inputs. Other external components required are clock resistor, integrator resistor and capacitor, and offset capacitor. Precision components are not required.

Of particular interest for the data acquisition systems are the display update (DU) and the end of conversion (EOC) pins. The EOC pin indicates the end of one conversion cycle and the start of the next conversion by a positive pulse one-half clock period long. The display update pin is an input to the chip which allows the data to be strobed into the output latches. If at least one positive edge is received prior to the ramp down cycle, new data is strobed to the display. In this A/D system, EOC is connected to DU.

Also of significance to the data acquisition network is the input polarity detection sequence for the MC14433. Polarity for the current conversion cycle is determined in the previous conversion cycle. Thus, if the polarity is reversed, a second conversion cycle must be made in order to obtain a correct measurement.

The MC14433 requires two power supplies. The total voltage across the device must not exceed 18 volts. Pin 13 is the reference level for the output circuitry of the MC14433. If this pin is tied to 0 volts, the BCD output, digit select, and EOC will swing from 0 volts to VDD. If, however, pin 13 is tied to VEE, the output swing will be from VEE to VDD.

The clock for the MC14433 is internal to the chip, requiring only a single external resistor to set the frequency. An external clock may be used by driving pin 10. The total conversion time for the MC14433 is approximately 16400 clock periods. This conversion time includes the auto-zero cycle and the unknown input measurement cycle. The clock frequency may be operated up to about 400 kHz producing a conversion time of 40 ms.

MPU

The microprocessor system used for this design is the M6800 system. Although an understanding of the MC6800 MPU and its instruction set is necessary for understanding the operation of the DAN, space does not permit a description of the microprocessor. For this information, consult the list of references at the end of the article. However, a description of the operation of the MC6821 is important and is presented. The MC6821 is a peripheral interface adapter designed to provide a parallel I/O port for the microprocessor system.

The PIA contains two essentially identical 8-bit interface ports. These ports (A side, B side) each contain three internal registers that include the data register which is the interface from the data bus to the A/D, the data direction register which programs each of the eight lines of the data register as either an input or an output, and the control register which, in addition to other functions, switches the data bus between the data register and the data direction register. Each port to the PIA contains two additional pins, CA1 and CA2, for interrupt capability and extra I/O lines. The functions of these lines are programmable with the remaining bits of the control register. Figure 3 shows a functional block of the MC6821 PIA.

Each PIA requires four address locations in memory. Two addresses access either of the two (A or B sides) data/data direction registers while the remaining two addresses access either of the two control registers. These addresses are decoded by the chip select and register select lines of the PIA which are connected to the MPU address bus. Selection between the data register and data direction register is made by programming a "1" or "0" in the third least significant bit of each control register. A logic "0" accesses the data direction register while a logic "1" accesses the data register.

By programming "0"s in the data direction register each corresponding line performs as an input, while "1"s in the data direction register make corresponding lines act as outputs. The eight lines may be intermixed between inputs and outputs by programming different combinations of "1"s and "0"s into the data direction register. At the beginning of the program the I/O configuration is programmed into the data direction register, after which the control register is programmed to select the data register for I/O operation.

8-CHANNEL DATA ACQUISITION NETWORK

Figures 4 and 5 are the flow diagram for the 8-channel data acquisition network. Figure 4 shows the basic operation of the program while Figure 5 provides more detail on the A/D conversion routine. The hardware required for the data acquisition is shown in Figure 8; as can be seen, it is fairly simple, consisting of the MC14433, MC1403 reference, MC14051 analog multiplexer, and an MC6821 PIA. The PIA is used as the interface between the microprocessor address and the data bus to the A/D. The microprocessor and associated memory are not shown due to a wide variety of forms possible depending upon the task that the total system is performing.

The reference for the MC14433 is an MC1403 bandgap reference which provides an output voltage of 2.5 volts. This voltage is divided down by the 20k ohms pot to the 2.000 volt reference required by the MC14433. If a 200 mV reference is used, full scale for the DAN will be 199.9 mV.

The analog multiplexing required to handle the eight input channels is provided by a MC14051B CMOS multiplexer. This device selects one of eight inputs with a 3-bit binary code. The device is capable of switching dual polarity (plus or minus inputs) with a single polarity control voltage.

The MC14433 BCD output and digit select outputs are connected to the B side of the PIA as shown in lines 20-27 of the software routine. These lines of the software are comment lines only and do not result in code for the microprocessor. The B side data register of the PIA is labeled throughout the program as PIA1BD while the control register is labeled PIA1BC. The control I/O lines (CB1) of the B side PIA connected to EOC of the MC14433.

The first executable instruction for the program is in line 45 and starts a section called PIA assembly. The PIA sets the A side data register as all outputs and the B side data register as all inputs. From there the program goes to the main program simulation which, as its name implies, is a simulation of the user's main program. At such time in the user's program that some analog information is required, the A/D converson subroutine starting in line 60 is executed. This routine synchronizes the program with the A/D conversion cycle and selects the first chanel to be measured. The program must be synchronized to the A/D since the A/D is continuously converting, and channel selection must be performed at the very beginning of the conversion cycle.

After the first channel to be measued is selected (channel 8), the routine was for two EOC pulses from the MC14433. Two EOC pulses are required to ensure that the polarity configuration of the MC14433 is compatible with current input voltage polarity. The MC14433 determines polarity in the previous conversion cycle. Once the second EOC pulse occurs, the MPU must demultiplex the BCD information and check for overrange for this channel. After this has been done, the next channel is selected and after two more EOC pulses occur, the BCD data is again demultiplexed, etc. This procedure is performed for all 8 channels after which the program returns from the A/D conversion subroutine to the main program. Here the A/D information may be used as desired.

Looking at the DEMUX software for the 8-channel data acquisition network in more detail, storage of the final results begins in memory location $0010. Each BCD character is stored in four LSBs of these memory locations. See figure 6 for explanation of data storage. Each of the eight channel readings requires four memory locations with the MSD occupying the first of each sequence of four memory locations. The index register is used to keep track of the next storage location for the BCD information. At the end of each channel's conversion cycle, the index register points to the MSD of that channel.

As mentioned previously, the multiplexed BCD data from the MC14433 is demultiplexed in the DEMUX routine. A "1" is placed in bit 4 of POINTR which is compared with the contents of PIA1BD. Upon finding DS1 in the true state, the lower four bits of PIA1BD (BCD data) are placed in the proper memory storage location. POINTR is then shifted to the left as each digit select occurs to look for the next successive digit select line.

After all four digits are placed in memory, the MSD is checked for overrange. If this condition occurs, an $F1 is placed in the MSD for this channel. Otherwise the half digit and polarity are decoded. Memory location TEST is now used as a temporary storage location to decode the polarity. The half digit is placed in the LSB of the MSD and negative polarity is indicated by placing a "1" in the MSB of the MSD.

The 8-channel DAN conversion time is approximately 320 ms with a 400 kHz clock frequency on the MC14433.

SAMPLE AND HOLD

The dual ramp A/D conversion process requires that the input to the A/D remain constant during the conversion cycle. If it does not, a sample and hold circuit must be used to ensure a constant input.

EXPANSION OF THE DAN

Expanding the data acquisition network can be done by adding more MC14051B analog multiplexers and modifying the software. Line 64 must be changed to reflect the increased number of channels.

REFERENCES

Aldridge,Don: "Analog-To-Digital Conversion Techniques with the M6800 Microprocessor System", AN757, Motorola Semiconductor Products Inc.

M6800 Microprocessor Applications Manual, Motorola Semiconductor Products Inc.

M6800 Microprocessor Programming Reference Manual, Motorola Semiconductor Products Inc.

MC6800, MC6821 Data Sheets, M6800 Microprocessor System Design Data, Motorola Semiconductor Products Inc.

MC1403/1503 Data Sheet, Motorola Semiconductor Products Inc.

MC14051B Data Sheet, Motorola Semiconductor Products Inc.

MC14433 Data Sheet, Motorola Semiconductor Products Inc.

Aldridge,Don: "Data Acquisition Networks with NMOS and CMOS" AN770, Motorola Semiconductor Products Inc.

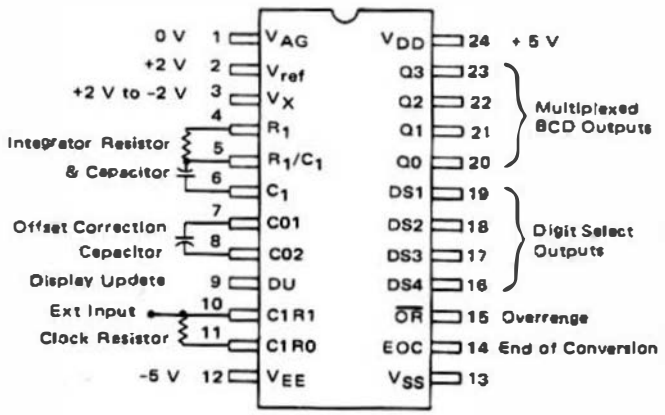


FIGURE 1 – MC14433 Pin Assignment

TRUTH TABLE

Coded Condition of MSD	Q3	Q2	Q1	Q0	BCD to 7 Segment Decoding
+0	1	1	1	0	Blank
−0	1	0	1	0	Blank
+0 UR	1	1	1	1	Blank
−0 UR	1	0	1	1	Blank
+1	0	1	0	0	4 → 1 Hook up
−1	0	0	0	0	0 → 1 only seg b
+1 OR	0	1	1	1	7 → 1 and c to
−1 OR	0	0	1	1	3 → 1 MSD

Notes for Truth Table
Q3 — ½ digit, low for "1", high for "0"
Q2 — Polarity: "1" = positive, "0" = negative
Q0 — Out of range condition exists if Q0 = 1. When used in conjunction with Q3 the type of out of range condition is indicated, i.e., Q3 = 0 → OR or Q3 = 1 → UR.

When only segment b and c of the decoder are connected to the ½ digit of the display, 4, 0, 7 and 3 appear as 1.

FIGURE 2 — MSD Coding

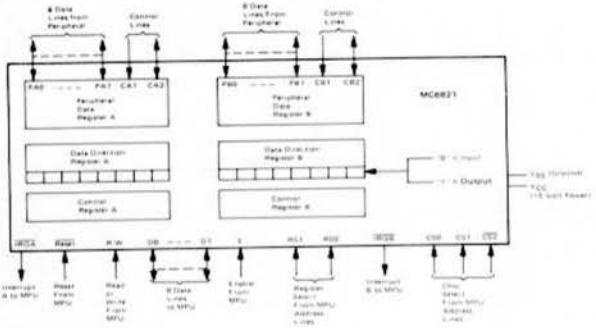


FIGURE 3 — PIA Functions

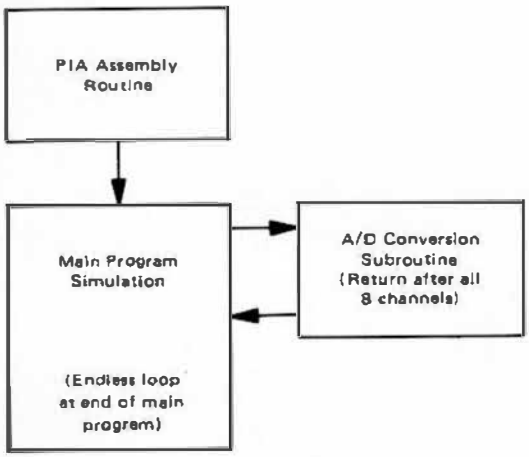
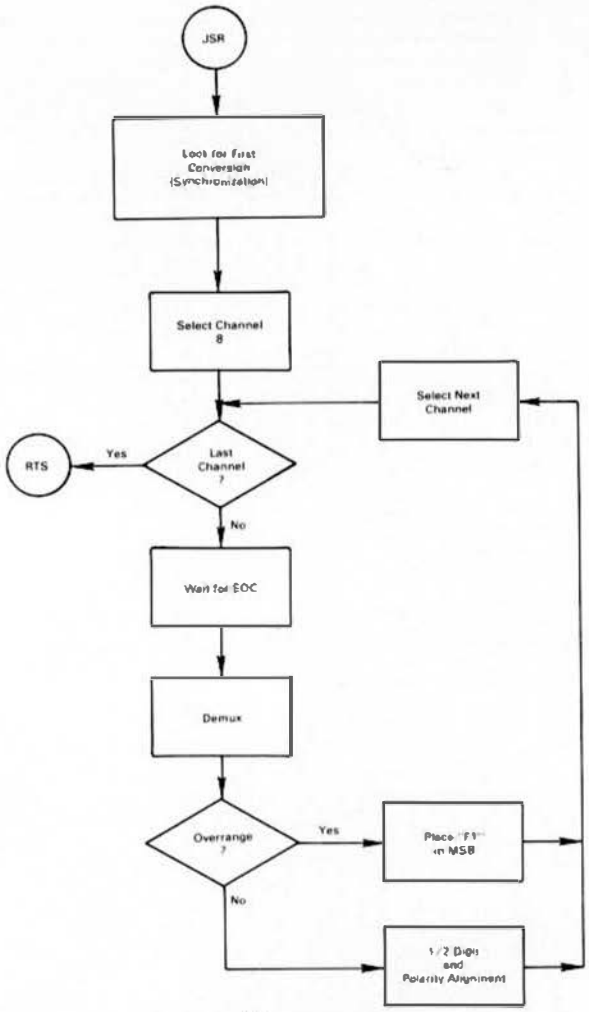
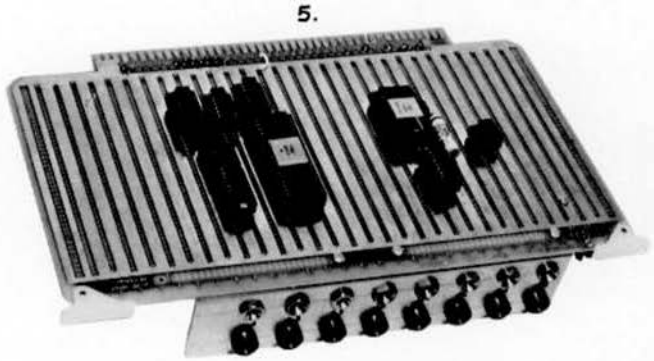


FIGURE 4 — Basic Operation of 8 Channel DAN



5.



8-CHANNEL DATA ACQUISITION NETWORK

Shown above is the 8-channel DAN discussed in this article. The board is Motorola EXORciser compatible. Extra IC packages shown on this board are for the required interface to the EXORciser.

The inputs for each channel are along top of the wirewrap board. The MC14433 A/D is the 24-pin package with its support components located adjacent to it.

7.

Channel Number	Memory Address	Digit	Data Example	Input Voltage
1	0010	MSD	01	1.729 V
	0011		07	
	0012		02	
	0013	LSD	09	
2	0014	MSD	F1	Overrange
	0015		09	
	0016		09	
	0017	LSD	09	
3	0018	MSD	08	-0.130 V
	0019		01	
	001A		03	
	001B	LSD	00	
4	001C	MSD	09	-1.130 V
	001D		01	
	001E		03	
	001F	LSD	00	
5	0020	MSD	00	0.000 V
	0021		00	
	0022		00	
	0023	LSD	00	
6	0024	MSD	01	1.000 V
	0025		00	
	0026		00	
	0027	LSD	00	
7	0028	MSD	F1	Overrange
	0029		09	
	002A		09	
	002B	LSD	09	
8	002C	MSD	09	-1.000 V
	002D		00	
	002E		00	
	002F	LSD	00	

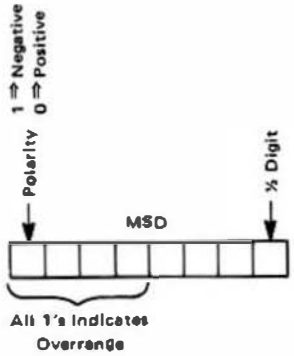


FIGURE 6 – Data Storage Definition

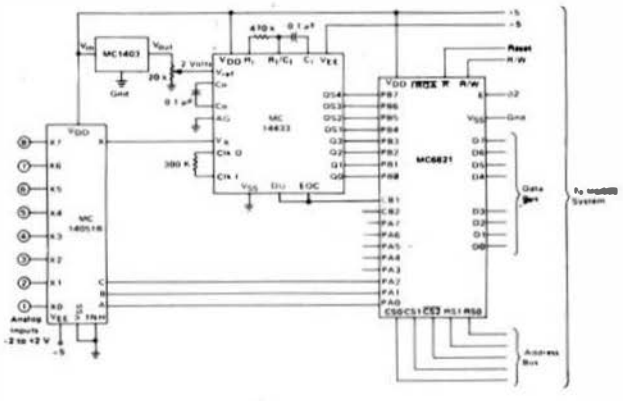


FIGURE 5 – 8-Channel Data Acquisition Hardware

```
                    MOTOROLA M6800AM CROSS-ASSEMBLER          PAGE  1

          M6800AM IS THE PROPERTY OF MOTOROLA SPD, INC.
            COPYRIGHT 1974 TO 1975 BY MOTOROLA INC

          MOTOROLA M6800 CROSS ASSEMBLER, RELEASE 1.2

00001                              NAM     LOGIC    GATE TESTER
00002       B004    PIAA    EQU     $B004    2.
00003       B006    PIAB    EQU     $B006    3.
00004       B005    PIACRA  EQU     $B005    4.
00005       B007    PIACRB  EQU     $B007    5.
00006       A01C    SCNCNT  EQU     $A01C    6.
00007       A00C    DISBUF  EQU     $A00C    7.
00008       B022    SCNREG  EQU     $B022    8.
00009       B020    DIGREG  EQU     $B020    9.
00010       A031    SIGN    EQU     $A031    10.
00011       A002    TEMP0   EQU     $A002    11.
00012       A004    TEMP1   EQU     $A004    12.
00013       A01E    CNT     EQU     $A01E    13.
00014 0000                  ORG     0        14.
00015 0000 7F08     TABLE   FDB     $7F08,$7F0E,$7F03,$0640,$600E   15.
      0002 7F0E
      0004 7F03
      0006 0640
      0008 600E
00016 000A 797F             FDB     $797F,$0E7F,$487F   16.
      000C 0E7F
      000E 487F
00017 0010 4F       INIT    CLR A            17.
00018 0011 B7 B005          STA A   PIACRA   18.
00019 0014 B7 B007          STA A   PIACRB   19.
00020 0017 B7 B006          STA A   PIAB     20.
00021 001A 86 FF            LDA A   #$FF     21.
00022 001C B7 B004          STA A   PIAA     22.
00023 001F 86 04            LDA A   #$04     23.
00024 0021 B7 B005          STA A   PIACRA   24.
00025 0024 B7 B007          STA A   PIACRB   25.
00026 0027 7F A004          CLR     TEMP1    26.
00 27 002A BD 27            BSR     CLEAR    27.
00028 002C BD 31            BSR     LED      28.
00029 002E 86 E0    KEY     LDA A   #$E0     29.
00030 0030 B7 B022          STA A   SCNREG   30.
00031 0033 7D B020          TST     DIGREG   31.
00032 0036 2A 04            BPL     TEST     32.
00033 0038 BD 41            BSR     DSPLY    33.
00034 003A 20 F2            BRA     KEY      34.
00035 003C BD 15    TEST    BSR     CLEAR    35.
00036 003F 4F               CLR A            36.
00037 0040 C6 04            LDA B   #$04     37.
00038 0042 F7 A01E          STA B   CNT      38.
00039 0044 BD 59    LOOP    BSR     BITEST   39.
00040 0046 7A 601E          DEC     CNT      40.
00041 0049 27 04            BEQ     END      41.

00042 004B BD 55            JSR A   #$55     42.
00043 004D 20 F5            BRA     LOOP     43.
00044 004F BD 0E    END     BSR     LED      44.
00045 0051 20 DA            BRA     KEY      45.
00046 0053 CE A031  CLEAR   LDX     #SIGN    46.
00047 0056 6F 00    NXT     CLR     0,X      47.
00048 0058 08               INX              48.
00049 0059 8C A047          CPX     #SIGN+A  49.
00050 005C 26 F8            BNE     NXT      50.
00051 005E 39               RTS              51.
00052 005F CE A00C  LED     LDX     #DISBUF  52.
00053 0062 A6 35    NXT1    LDA A   37,X     53.
00054 0064 FF A002          STX     TEMP0    54.
00055 0067 B7 A000          STA A   TEMP1+1  55.
00056 006A FE A004          LDX     TEMP1    56.
00057 006D A6 00            LDA A   0,X      57.
00 58 006F FE A002          LDX     TEMP0    58.
00059 0072 A7 00            STA A   0,X      59.
00060 0074 08               INX              60.
00061 0075 8C A012          CPX     #DISBUF+6   61.
00062 0078 26 E8            BNE     NXT1     62.
00063 007A 39               RTS              63.
00064 007B 86 20    DSPLY   LDA A   #$20     64.
00065 007D B7 A01C          STA A   SCNCNT   65.
00066 0080 CE A00C          LDX     #DISBUF  66.
00067 0083 A6 00    NEXT    LDA A   0,X      67.
00068 0085 B7 B020          STA A   DIGREG   68.
00069 0088 B6 A01C          LDA A   SCNCNT   69.
00070 008B B7 B022          STA A   S  REG   70.
00071 008E 86 50            LDA A   #$50     71.
00072 0090 4A       DELAY   DEC A            72.
00073 0091 26 FD            BNE     DELAY    73.
00074 0093 08               INX              74.
00075 0094 8C A012          CPX     #DISBUF+6   75.
00076 0097 27 05            BEQ     EXIT     76.
00077 0099 74 A01C          LSR     SCNCNT   77.
00078 009C 20 E5            BRA     NEXT     78.
00079 009E 39       EXIT    RTS              79.
00080 009F CE A031  BITEST  LDX     #SIGN    80.
00081 00A2 B7 B004          STA A   PIAA     81.
00082 00A5 F6 B006          LDA B   PIAB     82.
00083 00A8 54       LOOP1   LSR B            83.
00084 00A9 69 00            ROL     0,X      84.
00085 00AB 08               INX              85.
00 86 00AC 8C A037          CPX     #SIGN+6  86.
00087 00AF 26 F7            BNE     LOOP1    87.
00088 00B1 39               RTS              88.
00089                       END              89.


PAGE  001  MPUDAN

00001                       NAM     MPUDAN   DON ALDRIDGE
00002                  *
00003                  *
00004                  ********************************************************
00005                  *  8 CHANNEL DATA AQUISITION NETWORK WITH MC14422      *
00006                  *                WITH AUTO POLARITY                    *
00007                  ********************************************************
00008                  *
00009 0000                  ORG     $0000
00010 0000 0001     POINTR  RMB     1
00011 0001 0001     TEST    RMB     1
00012                  *
```

```
0000 0000                      ORG     $0000
0014 0000 0001    PIA1AD RMB   1       A SIDE DATA REGISTER
0015 0001 0001    PIA1AC RMB   1       A SIDE CONTROL REGISTER
0016 0002 0001    PIA1BD RMB   1       B SIDE DATA REGISTER
0017 0003 0001    PIA1BC RMB   1       B SIDE CONTROL REGISTER
0018                *
0019                *
0020                *          ** PIA CONFIGURATION **
0021                *.....
0022                * PA7 * PA6 * PA5 * PA4 * PA3 * PA2 * PA1 * PA0 *
0023                *****************************************
0024                * LSD                 MSD + MSD           LSD+
0025                *****************************************
0026                *     DIGIT SELECT    +        BCD        +
0027                *.....
0028                *
0029                *        RESULTS STORED IN LOCATIONS $0010-$002F
0030                *           EACH CHANNEL OCCUPIES FOUR CONSECUTIVE
0031                *           MEMORY LOCATIONS WITH MSD FIRST
0032                *           NEGATIVE POLARITY INDICATED VIA A
0033                *           "1" IN MSB OF THE MSD
0034                *           OVERRANGE INDICATION VIA AN "F1" IN
0035                *           MSD OF EACH CHANNEL
0036                *
0037                *           CHANNEL SELECTION VIA PIA1AD
0038                *           CHANNEL NUMBER IS CODED IN A BINARY
0039                *           FORM FOR CHANNELS 0-7
0040                *
0041                *
0042                *
0043 0200                      ORG     $200    PIA INITIALIZATION
0044 0200 7F 0001              CLR     TEST
0045 0203 7F 0003              CLR     PIA1BC
0046 0206 7F 0001              CLR     PIA1AC
0047 0209 CE 0006              LDX     #$0006
0048 020C FF 0002              STX     PIA1BD
0049 020F CE FF06              LDX     #$FF06
0050 0212 FF 0000              STX     PIA1AD
0051                *
0052                *
0053 0215 01                   NOP             MAIN PROGRAM SIMULATION
0054 0216 BD 021C              JSR     CONVRT
0055 0219 01          END      NOP
0056 021A 20 FD                BRA     END
0057                *
0058                *
0059 021C CE 000C     CONVRT   LDX     #$000C  CONVERSION ROUTINE
0060 021F 7D 0002              TST     PIA1BD  CLEAR PIA FLAGS
0061 0222 B6 0002     WAIT1    LDA A   PIA1BC
0062 0225 2A FB                BPL     WAIT1
0063 0227 7D 0002              TST     PIA1BD
0064 022A C6 07                LDA B   #$07    SELECT CHANNEL 0
0065 022C F7 0000     N        STA B   PIA1AD
0066 022F 7D 0003     WAIT2    TST     PIA1BC  FIRST CONVERSION COMPLETE?
0067 0232 2A FB                BPL     WAIT2
0068 0234 7D 0002              TST     PIA1BD  CLEAR PIA FLAGS
0069 0237 B6 0001     WAIT3    LDA A   PIA1BC  SECOND CONVERSION COMPLETE?
0070 023A 2A FB                BPL     WAIT3
0071                *
0072                *
0073 023C 86 10       DEMUX    LDA A   #$10    INITIALIZE DIGIT SEL POINTER
0074 023E 97 00                STA A   POINTR
0075 0240 B6 0002     NEXT     LDA A   PIA1BD
0076 0243 16                   TAB
0077 0244 94 00                AND A   POINTR  LOOK FOR DIGIT SELECT
0078 0246 27 F8                BEQ     NEXT
0079 0248 78 0000              ASL     POINTR  SELECT NEXT DIGIT
0080 024B C4 0F                AND B   #$0F    SAVE BCD DATA
0081 024D E7 04                STA B   4,X     STORE BCD DATA IN MEMORY
0082 024F 08                   INX
0083 0250 24 EE                BCC     NEXT    LAST DIGIT?
0084 0252 A6 00                LDA A   0,X     OVERRANGE TEST
0085 0254 16                   TAB
0086 0255 84 0B                AND A   #$0B
0087 0257 81 03                CMP A   #$03
0088 0259 27 14                BEQ     OVFMOD
0089 025B 7F 0001              CLR     TEST    HALF DIGIT AND POLARITY
0090 025E C4 0C                AND B   #$0C    ALIGNMENT
0091 0260 54                   LSR B
0092 0261 54                   LSR B
0093 0262 54                   LSR B
0094 0263 76 0001              ROR     TEST
0095 0266 00 01                ADD B   TEST
0096 0268 52                   COM B
0097 0269 C4 01                AND B   #$01
0098 026B E7 00                STA B   0,X
0099 026D 20 04                BRA     FINE
0100 026F 86 F1       OVFMOD   LDA A   #$F1    OVERRANGE ROUTINE
0101 0271 A7 00                STA A   0,X
0102 0273 7A 0000     FINE     DEC     PIA1AD  SELECT NEXT CHANNEL
0103 0276 2A B7                BPL     WAIT2   LAST CHANNEL?
0104 0278 39                   RTS
0105                *
0106       0000                END

TOTAL ERRORS 00000
```

LOGIC GATE TESTER

S. J. Houng
E. 36 Salmon St.
Spokane, WA 99218

This program provides a dynamic testing for the following integrated circuits (ICs):

(1) Logic gates - NAND, NOR, AND, OR, and Exclusive-OR gates,

(2) Buffers and Inverters,

(3) Flip-Flops and Bistable Latches.

The number of gate inputs can be 1,2,3,4, or 8. Six gates can be tested at one time. The testing is a one-step operation. By pushing the Go key, the computer will analyze the output response of each gate, and tell you which gate is good, and what kind of gate it is.

The testing procedure is as follows

(1) The computer sends a sequence of one byte test pattern to the input of the gate in the form of 00,55,AA and FF hexadecimal.

(2) For each test pattern, the computer reads the output response of each gate, and compares it with the Truth Table in Table 1.

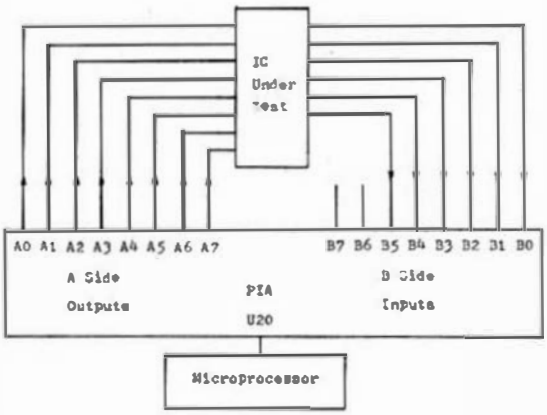
(3) After the Table has been matched with the output response, the computer will display a message on 6 7-segment LEDs (Light Emitting Diode). Each LED represents a gate. The character displayed on the LED represents the type of gate.

The required hardware are a microprocessor, a peripheral interface adapter (PIA), and an experimental IC socket, as shown in Figure 1. On the PIA, the A side data lines, A0-A7, are programmed as outputs, and the B side data lines, B0-B7, as inputs. Following the closure of the Go key, a sequence of test patterns are placed on data lines A0-A7, which also represent the inputs of the gate under test. On the B side, the gate output responses are read by data lines B0-B5 (B6 and B7 are not used). Each output is converted into a binary value and compared with the Truth Table in Table 1. The result is displayed as a special character on the LED, to represent the type of gate. If the character agrees with the type of gate under test, you have a good gate, otherwise a bad one. The test circuit connections for each type of logic gate, and flip-flop, are given in Figure 2.

The program in Listing 1 is written for the Motorola MEK6800D2 kit. The special gate character for the Truth Table occupy the memory locations from 00 to 0F hexadecimal. Starting from the hexadecimal address 10, the PIA is initialized to have outputs on the A side, and inputs on the B side. The LED display is then blanked. Following the PIA initialization, the keyboard is tested by the routine KEY to determine whether or not the Go key has been pressed. If it has been closed, the TEST routine is carried out. After each test, the test pattern is increased by a number of 55 hexadecimal. BITEST then stores the bit output from each gate at its own memory location, beginning at the address SIGN. At the same time, it shifts the previous bit output to the left of its memory. Following the TEST, the LED routine converts the binary value, stored at SIGN, into the gate character, and stores it at DISBUF locations. Finally, the DSPLY routine displays 6 characters in LEDs on the keyboard. The displayed message will tell the whole story about the logic gate.

Test Pattern	Inverter	Buffer
0	1	0
1	0	1
0	1	0
1	0	1
Hex. Value	A	5
Displayed Character		

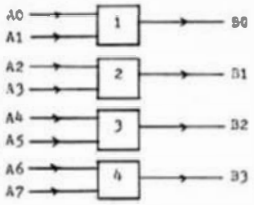
Clock	D-Input	D-Type Flip-Flop Q	D-Type Flip-Flop $\bar{Q}$	J-K Flip-Flop J=K=1 Q	J-K Flip-Flop J=K=1 $\bar{Q}$
0	0	1	0	0	1
1	0	0	1	0	1
0	1	0	1	1	0
1	1	1	0	1	0
Hex. Value		9	6	3	C
Displayed Character					



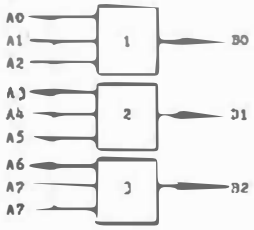
Sequence	Test Pattern A0	A1	A2	A3	A4	A5	A6	A7	Hex. Value
1	0	0	0	0	0	0	0	0	00
2	1	0	1	0	1	0	1	0	55
3	0	1	0	1	0	1	0	1	AA
4	1	1	1	1	1	1	1	1	FF

1.

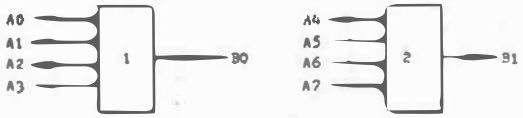
(a) Quadruple 2-Input Gate



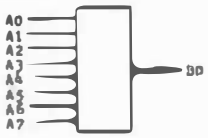
(b) Triple 3-Input Gate



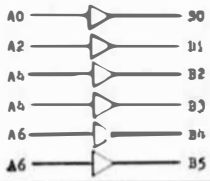
(c) Dual 4-Input Gate



(d) 8-Input Gate



(e) Hex Buffers / Inverters



(f) Flip-Flops / Latches

Terry "Tex" Ritter
12002B Pollyanna
Austin, TX 78753

PROGRAMMING THE 6809

Now that systems using the Motorola MC6809 processor are becoming available, programmers are starting to think and write in 6809 code. Unfortunately, some of the most-important features of the machine seem to have been lost on the very people who should have used them first. Although we deliberately designed the '09 to be source code compatible with the '00 (to take advantage of the existing base of software and training), even well-organized 6800 code is rarely optimal for the 6809.

Previous first (8008), second (8080, 6800), and second-and-a-half (Z-80, 8085) generation processors did not allow programming techniques which are useful and easy on the 6809. Obviously, any program which "works" has some value; but in developing a personal computer environment, it makes sense to develop routines which can be broadly re-used, and an environment which does not restrict future development. Restrictions on future development can occur almost incidentally--one only need investigate DMA operation on the "industry standard" S100 bus, to see a significant advantage of the sparser but better-defined S550. But a personal computer is even more influenced by resident software than the hardware environment. And it is important to use the best designs we can, to minimize the somewhat inevitable re-do projects.

Upward Compatibility

It is very easy for software houses to convert their 6800 code to the 6809 code, since the 6809 includes the 6800 architecture as a proper subset, and since the Motorola 6809 macroassembler fully accepts the old 6800 mnemonics. Nevertheless, a relatively-small amount of work on the original code will produce a better finished product, one that will deliver truly improved performance on the 6809 (most "Z-80" code never used the features of the Z-80, and thus most code performed virtually as well on an 8080). Because the consumer will purchase new code, he has a right to demand aspects of programming which were not feasible on older processors. And, if he looks at "6809 source code," he has every reason to expect to see the logically-consistent 6809 mnemonics, and not their older 6800 equivalents.

Aspects of Quality

Some important aspects of programming which are no-longer unreasonable are modular, re-entrant, and position-independent programming.

Position-Independence. A program is said to be "position-independent" when it will run correctly when the exact same machine code is positioned arbitrarily in memory. Such a program is useful in many different hardware configurations, and might be copied from a disc into RAM when the Operating System first sees a request to use that system utility. Position-independent programs rarely use absolute addressing; instead indexed and relative modes are used heavily. In particular, there should be no JMP <absolute> or JSR <absolute> instructions at all (each JMP or JSR should be replaced by an LBRA or LBSR, respectively, requiring no more code); although position-independence does incur some run-time overhead, this is almost always preferable to a position-dependent program. When pointing at constants or tables within the program (consider, for example, text-strings for various console messages), the programmer should use the "program-counter-relative" (PCR) form of indexed addressing (which is position-independent), instead of loading an absolute address (which is not).

Modular Programming. Programming in modules is another indication of quality code, whether for the 6809 or any other machine (unfortunately, modular code is really tough on any early-generation micro). A module is a program-element which can be easily disconnected from the rest of the program, either for re-use in a new environment, or for replacement (due to error in the original). A module is usually a subroutine (although a subroutine is not necessarily a module); frequently, the programmer isolates register-changes internal to the module from the calling routine (on the 6809 one would PSH all registers changed by the module, do the work, then PUL those same register values back, plus PC, thus replacing the usual RTS). Isolating register changes in the called module to that module alone allows the code in the calling program to be more-easily analyzed since it can be assumed that all registers (except those specifically used for parameter transfer) are unchanged by each called module. This leaves the CPU registers free at each level for loop-counts, address-comparisons, etc.

Local Storage. A clean method for allocating "local" storage is required both by position-independent programs as well as modular code--local (temporary) storage is used to hold values only during execution of a module (or called modules) and is released upon return. Almost always a module will need more temporary storage than just the CPU registers; temporary storage is readily available from the

stack area on the 6809. The desired amount of storage is allocated by moving the hardware stack pointer down (LEAS -n,S) as desired, then back again at the end of the routine (LEAS n,S); values stored in this area are easily accessed by indexing from the stack pointer (e.g., ADD# m,S). Naturally, parameters can be passed from and back to the caller either in registers or on the stack, and although the stack technique does involve more overhead, it is also extensible to n parameters of length m; an amount of storage which could never be provided by the registers in any CPU.

Global Storage. Even in a modular environment there is need for "globals;" values which are accessible by many modules within a given system. These provide a convenient means for storing values from one invocation to another invocation of the same routine. One appropriately position-independent global technique is to locate the U-register at the present top-of-stack, then move the stack pointer down past the space reserved for globals. By convention, all modules will preserve that same value for U when calling subordinate modules. Modules at all levels have identical access to the globals, by performing accesses indexed from the U-register (e.g., LDD cat,U).

The Interface Level

Granted that we have a well-designed program which uses modern programming practices, the position-independent module must still interface into hardware with some fixed characteristics which vary between systems. In fact, it must also interface the same system as it develops over time.

Don't Paint Yourself In. Software tends to change through time. Since later (corrected) versions of the same module may well have routines positioned differently, it makes sense to list all externally-useful modules in a table in the beginning of the overall module; typically, a list of LBRAs to the internal modules will be sufficient (until further standards are finished for automatically linking internal and external routines).

Allow Different I/O. Hardware tends to change through time. Furthermore, different systems have different hardware (e.g., there is currently a boom in memory-mapped video boards--hopefully using the industry-standard MC6845 CRT Controller--which will indeed replace the RS-232 terminal in those systems). The 6809 includes indexed-indirect addressing modes deliberately-designed to simplify this problem. The core module can be built to expect both I/O routine locations, and I/O device locations as parameters passed to it on the stack. If the U-register is positioned and global area allocated upon entry, all internal modules can access I/O routines or devices by indexing the correct parameter from the U-register, then accessing through that pointer. One would thus expect that the first entry in the long-branch table would be a small set-up routine to configure the core module for the present environment; The next entry would be to the core module itself, thus allowing external routines to re-configure the system and re-invoke it.

The System Environment

Just as interactive computer operation is far preferable to batch (still used by many companies and even CS schools, unfortunately), multi-tasking capability is much preferable to the usual single-job-at-a-time system organization. Additionally, there is actual need for an interrupt-driven real-time environment to support a large number of computationally trivial tasks around the home. These desired goals are supported best by modular, re-entrant code, using the same tech-

niques as described above, plus the small matter of a multi-tasking real-time Operating System.

The OS. An operating system is the heart of a computer in the sense that it supports necessary operations (e.g., I/O, disc access, etc.) in such ways that support the desired re-entrant multi-task capabilities and relieve that support from the user programs. In my opinion, no program deserves the title "Operating System" unless it does allocation of scarce CPU resources (e.g. memory, storage, CPU time, I/O channels, etc.). Programs which enable the user to enter and execute machine code are frequently called "debug monitors" on microcomputers. These are all we will have for a while. But getting or writing a monitor with capabilities largely typical of those used on previous-generation micros should not blind us of our eventual goals: the real-time multi-tasking computer of the home.

Conclusion

A number of software techniques have been described, both to jog programmers into new opportunities available on the 6809, and to give consumers some indications of some things which are possible in quality 6809 code. Examples of some of these techniques have been in the public domain for some time, see my "Resident Memory Test Systems with an Example for the 6800," (sic) in Dr. Dobb's Journal of Computer Calisthenics and Orthodontia, Vol. 3 No. 5, May 1978.

★ SOFTWARE CONTEST ★
$12,000.00+
AND GROWING

68 MICRO JOURNAL SOFTWARE CONTEST

Prize list now at $12,000.00+ and growing.

Prizes for each category will be:

FIRST - Life Subscription 68 Micro Journal™

SECOND - 6 year extension 68 Micro Journal™

THIRD - 3 year extension 68 Micro Journal™

4th-10th 1 year extension 68 Micro Journal™

The software must be utilities or serious software, of original design, to operate with the following CATEGORIES:

TSC FLEX	6800 Disk System MiniFlex
TSC FLEX	6800 Disk System FLEX Ver. 2.0 5"
TSC FLEX	6800 Disk System Ver. 1.0 8"
TSC FLEX	6809 Disk System Ver. 09 5" or 8"
SSB	Dos Version 5
SSB	Dos Version 4 or earlier
SSB	TSC FLEX Version
PERCOM	INDEX
PERCOM	MiniDos+
HEMENWAY	CP/68 Disk System
MSI	Disk Operating Systems
SOFTWARE DYNAMICS	SOOS
JPC	TC-3 Cassette System
ANY	KC Standard Tape System
BASIC	Any 6800 Version
BASIC	Any 6809 Version

There are sixteen (16) categories, as indicated above. In addition we have other prizes donated by various vendors of 6800/09 products. As of the 15th of October 1979 over $12,000.00 in prizes has been pledged.

GIMIX - Mainframe, value $798.19.

SOUTHWEST TECHNICAL PRODUCTS CORP (SWTPC) Computer 69/8, value $595.00.

DIGITAL RESEARCH: COMPUTERS, 16K static memory board kit, value $295.00.

JPC PRODUCTS, TC-3 kit w/CFM-3 package; AD-16 kit; CK7 kit, total value $190.00.

STAR-KITS, Set checkbook balancing software : MiniFLEX™ or PERCOM Super BASIC on disk, value $40.00.

MICROWARE Systems Corp, Package 1 each: ABASIC Compiler, ABASIC Source Gen., ABASIC Interpreter, LISP Interpreter; also 6 each RT/68 ROM OS and 6 each 6800 Chess programs. Total value $1,005.00.

SMOKE SIGNAL BROADCASTING, two $500.00 gift certificates for any SSB product. Categories eligible SSB DOS Ver. 4 or earlier and Ver. 5 or later; total value $1,000.00.

LUCIDATA, 5 each PASCAL Ver. 2, 1 to each FLEX™ category. Total value $750.00.

COMPUTERWARE, 2 each $200.00 gift certificates for any Computerware software product. 1 gift certificate for best of SSB disk BASIC and 1 gift certificate for best of Computerware's Random BASIC. Total value, $400.00.

SSI, Schreier Software Index, choice of $100.00 of SSI software.

SOFTWARE DYNAMICS, 6800 BASIC Compiler, value $210.00.

The MICRO WORKS, choice of any one item, value up to $179.95.

HEMENWAY ASSOCIATES, INC., Books, 3 each CP/68, 3 each XA6809 Macro Cross Assembler, 3 each STRUBAL+ Compiler, total value $329.55.

CER-COMP Microcomputers, MiniDisk+ Disk system (EPROM) and software on disk. Total value $89.00.

TECHNICAL SYSTEMS CONSULTANTS (TSC), $250.00 choice of any TSC software, for best of FLEX™ entries.

Final decision shall be delegated to a panel of judges selected by the staff of 68 Micro Journal™. All judges decisions are final and each person submitting, shall by his or her submitting material for evaluation, acknowledge that they agree to abide by any and all rules of this contest, as published within the pages of 68 Micro Journal™.

Programs and material submitted shall be judged on the basis of good and workable software. By this we mean, it should do something useful and be needed by the average 6800/09 user in the particular category. Size is of little importance, the most important consideration will be how useful it is.

All material submitted shall remain the property of the original owner (who should be the author). Each submission shall contain a paragraph that states the material submitted is of original design and the property of the person in whose name it is submitted.

It shall be understood that regardless of who wins or does not win a prize, all material submitted shall be authorized and eligible, to be published by 68 Micro Journal™. Material published, which was not a winning entry, shall gain the author an extension to his or her subscription. Anyone may enter and it is not a requirement that the person submitting material be a subscriber to 68 Micro Journal™. Prizes will be awarded on the quality of the material submitted and being or not being a subscriber, will have no bearing.

Authors should indicate that the material has NOT been previously published in any commercial magazine or journal (club newsletters and the like do not count as a commercial magazine or journal).

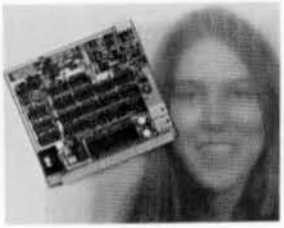
I have tried to keep the rules simple. This should encourage the maximum participation in the contest. This is another of the ways that we attempt to secure good material for the sole benefit of our readers. Also I believe that it will encourage those who have developed good software, to share with his or her fellow 6800/09 users. By sharing we all profit. By working together, as has been in the past, it enables us as 6800/09 user to have a magazine that is just for us.

This contest was scheduled to close February 15th, 1980. This will be the first anniversary of 68 Micro Journal™. Prize winners will be announced in the April or July 1980 issue of 68 Micro Journal™.

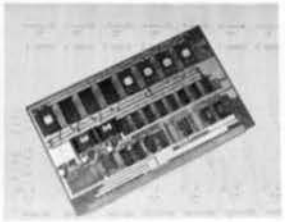
NOTE: Due to the large amount of material being submitted, the contest may be extended for an additional 90 days (February to May). The prime consideration of this contest is not time but good software. By this extension we can include some excellent software, nearing completion, that we would otherwise miss making available to our readers.

ALL ENTRIES NOTIFIED AS FINAL PRIZE WINNING ENTRIES WILL BE REQUIRED TO SUBMIT MATERIAL (IN SOURCE AND BINARY) ON MEDIA OF CATEGORY USED. This will allow the judges to assemble and run all winning programs, as submitted.

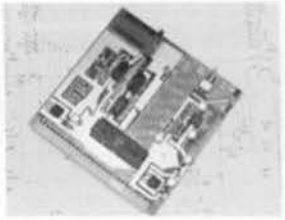
All entries should be plainly marked 'CONTEST'!!!



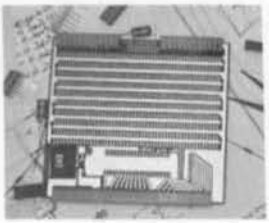
DS-68 DIGISECTOR



PSB-08 PROM SYSTEM BOARD



B-08 EPROM PROGRAMMER



UIO UNIVERSAL I/O BOARD

INNOVATIVE PRODUCTS FOR 6800 USERS

DS-68 DIGISECTOR is a random access video digitizer featuring 256 × 256 picture element scan and 64 levels of grey scale, with conversion times as low as 3 microseconds per pixel. It accepts either interlaced (NTSC) or non-interlaced (industrial) video input. Use it for computer portraiture, moving target indicators, precision security systems, fast to slow scan conversion... with clever software, the Digisector can read just about anything. Truly a professional tool at a price you can afford. $169.95

DM-85 DISK MIXER is an add-on board for the Smoke Signal Broadcasting BFD-68A Disk Controller which allows operation of both 8" and 5" drives. Controller mode (8" or 5") is selected on a drive-by-drive basis, so any mix of 5" and 8" drives is allowable. The 2" × 3" PC board mounts inconspicously on the back of the BFD-68A. Its operation is completely transparent to software. An oscilloscope is required for the setup procedure. Kit Price: $39.95

B-08 2708 EPROM PROGRAMMER is a compact unit that fits in the 6800's I/O slot. A safety switch and LED indicator provide control over the high programming voltage generated on board. An industrial quality Textool socket and extended board height allow effortless PROM insertion and retrieval. Fully commented source listings of U2708 is included in the Owner's Manual. $99.95

U2708 utility for testing, burning, verifying and copying 2708s in EPROM. $29.95

M6809 EMULATOR is a machine language program that will emulate all of the functions of the Motorola 6809 third generation microprocessor. Developed for use on any 6800 system, the program allows software development and debugging. The 3K byte program is complete with a 6809 mini-monitor and single-step trace routines. Fully commented source listing included. Specify Smoke Signal Broadcasting or FLEX™ disk, or KCS cassette. $49.95

PSB-08 PROM SYSTEM BOARD features 1K of high speed, low-power RAM and space for up to 8 2708 EPROMs, both DIP-switch addressable to start on any 8K boundary in memory. The exclusive I/O select feature allows you to move I/O locations up to any unused 1K block in the EPROM memory space. This permits memory expansion to a full 56K of contiguous user RAM. $119.95

UIO UNIVERSAL I/O BOARD helps you with your custom interfaces. It has space for a 40-pin wire wrap socket into which you may plug any of Motorola's 40 or 24-pin interface chips. All data and control lines are connected to the appropriate edge connector pins. All other bus connections are brought out to a 16-pin socket pad. +5 volt regulator and all Molex connectors are provided; regulated +5 and ground are bused among the locations for up to 35 14-pin ICs. $24.95

THE MICRO WORKS **P.O. BOX 1110, DEL MAR, CA 92014 714-942-2400**

NEW PRODUCT

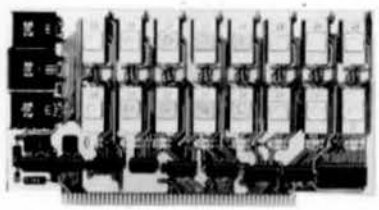
Jim Schreier of SSI, The Schreier Software Index, announces the addition of FLEX2™ applications programs to their growing software stock.

Among the new offerings are the following: 68 of the 76 programs in Poole's and Borcher's 'Some Common BASIC Programs' (2 Disk); an Income Expense Ledger allowing 18 expense accounts and 8 income accounts; a complete Genealogy Program allowing up to 5 generations per disk; and a Professional Rodeo Cowboy's System allowing calculations of Entry Fees and Rodeo Payoffs, as well as Rodeo Bookeeping (PRCA).

The above software items are available for the 5" minidisk only. Price per disk is $16.95. A 3 ring plastic notebook holder with documentation is included. Additional information can be secured from:

SSI, The Schreier Software Index, 4327 East Grove Street, Phoenix, AZ 85040.

16K EPROM CARD-S 100 BUSS



$59.95 KIT

OUR BEST SELLING KIT!

USES 2708's!

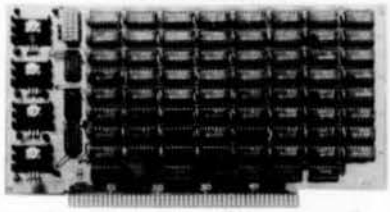
Thousands of personal and business systems around the world use this board with complete satisfaction. Puts 16K of software on line at **ALL TIMES!** Kit features a top quality soldermasked and silk-screened PC board and first run parts and sockets. All parts (except 2708's) are included. Any number of EPROM locations may be disabled to avoid any memory conflicts. Fully buffered and has WAIT STATE capabilities.

OUR 450NS 2708'S ARE $8.95 EA. WITH PURCHASE OF KIT

ASSEMBLED AND FULLY TESTED ADD $25

8K LOW POWER RAM KIT-S 100 BUSS

SALE



PRICE CUT!

$119.50 KIT

(450 NS RAMS!)

Thousands of computer systems rely on this rugged, work horse, RAM board. Designed for error-free, NO HASSLE, systems use.

KIT FEATURES:

1. Doubled sided PC Board with solder mask and silk screen layout. Gold plated contact fingers.
2. All sockets included.
3. Fully buffered on all address and data lines.
4. Phantom is jumper selectable to pin 67.
5. FOUR 7805 regulators are provided on card.

Blank PC Board w/Documentation **$29.95**
Low Profile Socket Set...**13.50**
Support IC's (TTL & Regulators) **$9.75**
Bypass CAP's (Disc & Tantalums) **$4.50**

ASSEMBLED AND FULLY BURNED IN ADD $30

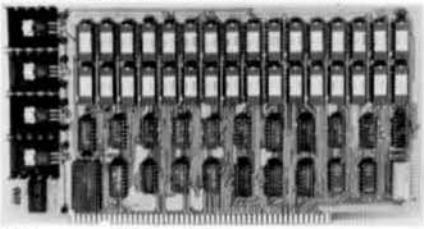
16K STATIC RAM KIT-S 100 BUSS

PRICE CUT!

$279 KIT

FOR 250NS ADD $10

FULLY STATIC, AT DYNAMIC PRICES



WHY THE 2114 RAM CHIP?

We feel the 2114 will be the next industry standard RAM chip (like the 2102 was). This means price, availability, and quality will all be good! Next, the 2114 is FULLY STATIC! We feel this is the **ONLY** way to go on the S-100 Buss! We've all heard the HORROR stories about some Dynamic Ram Boards having trouble with DMA and FLOPPY DISC DRIVES. Who needs these kinds of problems? And finally, even among other 4K Static RAM's the 2114 stands out! Not all 4K static Rams are created equal! Some of the other 4K's have **clocked** chip enable lines and various timing windows **just as critical** as Dynamic RAM's. Some of our competitor's 16K boards use these "tricky" devices. But not us! The 2114 is the ONLY logical choice for a trouble-free, straightforward design.

KIT FEATURES:
1. Addressable as four separate 4K Blocks.
2. ON BOARD BANK SELECT circuitry. (Cromemco Standard!). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams.
4. ON BOARD SELECTABLE WAIT STATES.
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers.
6. All address and data lines fully buffered.
7. Kit includes ALL parts and sockets.
8. PHANTOM is jumpered to PIN 67.
9. LOW POWER: under 2 amps TYPICAL from the +8 Volt Buss.
10. Blank PC Board can be populated as any multiple of 4K.

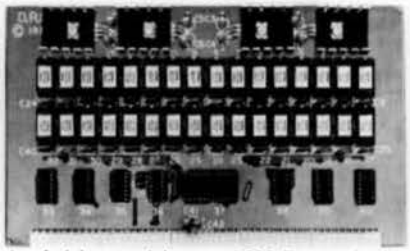
BLANK PC BOARD W/DATA—**$33**
LOW PROFILE SOCKET SET—**$12**
SUPPORT IC'S & CAPS—**$19.95**
ASSEMBLED & TESTED—**ADD $30**

16K STATIC RAM SS-50 BUSS

PRICE CUT!

$275 KIT

FULLY STATIC AT DYNAMIC PRICES



KIT FEATURES:

1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Runs at Full Speed
4. Double sided PC Board. Solder mask and silk screened layout. Gold fingers.
5. All Parts and Sockets included
6. Low Power: Under 2 Amps Typical

FOR SWTPC 6800 BUSS!

ASSEMBLED AND TESTED - $30

BLANK PC BOARD—**$26**
COMPLETE SOCKET SET—**$12**
SUPPORT IC'S AND CAPS—**$19.95**

S-100 Z80 CPU CARD

ASSEMBLED AND TESTED! READY TO USE! Over 3 years of design efforts were required to produce a **TRUE** S-100 Z80 CPU at a genuinely bargain price! **4 MHZ! $159.95**

FEATURES:

- 2 or 4 MHZ Operation.
- Generates MWRITE, so no front panel required.
- Jump on reset capability
- 8080 Signals emulated for S-100 compatability.
- Top Quality PCB, Silk Screened, Solder Masked, Gold Plated Contact Fingers.

Perfect For OEM's

LOW POWER - 250NS 2114 RAM SALE!

4K STATIC RAM'S. MAJOR BRAND, NEW PARTS. These are the most sought after 2114's, LOW POWER and 250NS FAST.

SPECIAL SALE: $7.50 ea. or 8 For $55

(We reserve the right to limit quantities.)

PROC. TECH. QUITS THE MICROPROCESSOR BUSINESS!
FACTORY CLOSE OUT - SPECIAL PURCHASE!
#16KRA

16K S-100 Dynamic Ram Board - $149.95

ORIGINALLY PRICED AT $429 each!

We purchased the remaining inventory of PT's popular 16K Ram Board when they recently closed their plant. Don't miss the boat! These are brand new, fully tested, ASSEMBLED and ready to go. All are sold with our standard 90 day limited warranty!!

72 Page Full Manual, Included Free!

NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.

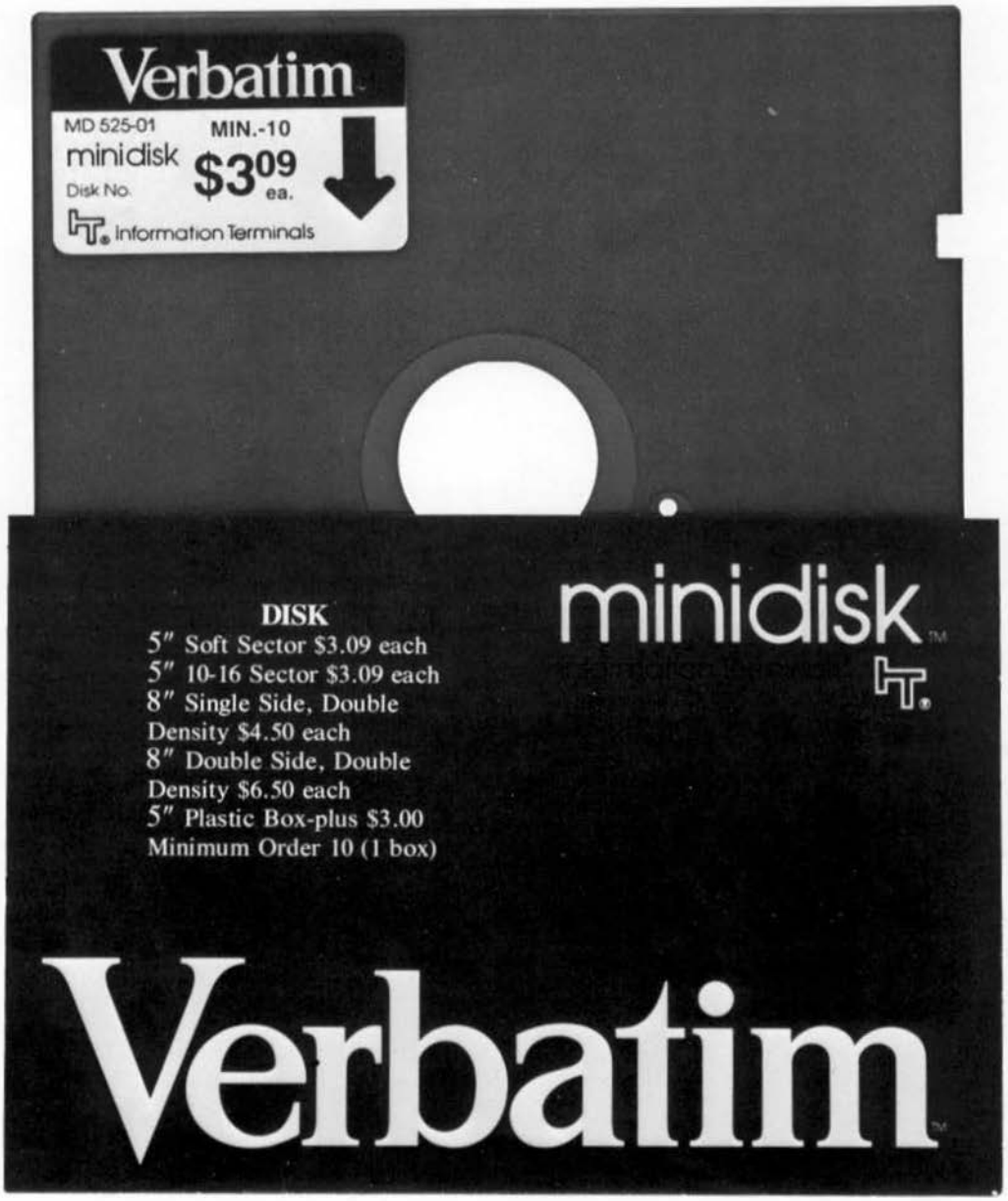
Digital Research: Computers
(OF TEXAS)

P.O. Box 401565 • GARLAND, TEXAS 75040 • (214) 494-1505

TERMS: Add $1.00 postage, we pay balance. Orders under $15 add 75¢ handling No C.O.D. We accept Visa, MasterCharge, and American Express cards. Tex. Res. add 5% Tax. Foreign orders (except Canada add 20% P & H 90 Day Money Back Guarantee on all items.

5-1/4" Minidisk — Soft or Hard Sector

SAVE



DISK

SOUTH EAST MEDIA SUPPLY

6131 Airways Blvd. 615-892-1328
Chattanooga, TN 37421

H H H Enterprises
Box 493, Laurel, Md.
20810 301-953-1155

SUPER SYSTEMS

SUPER SYSTEMS come in almost any size and consist of:

1) Super box, 32k ram, dual 5" floppies, an ADM-3A terminal, Smartbug, Random basic, text editor, assembler, documentation and cables. All for $3,840.

2) Super Box, 48k ram, dual 8" floppies, Random basic, text ed., assembler, word processor, Anadex DP-8000 printer, documentation and cables. All for $5,900.

3) "DELUX" (the really big system); Super box, 128k software controll ram, 4ea double sided, double density, 8" disk drives, Spinwritter printer, color terminal (80X48 lines, color graphics, etc.), Random basic, Random compiler, Fortran, Spirit, Pascal, A/BASIC family, 4 serial ports, 8 par. ports, and all for only $16,500.

All systems are tested and integrated at Laurel, Md. They are then demonstrated and/or packed for shipment. All software is setup for proper port operation and compatability.

SPECIAL! SPECIAL! SPECIAL!

A limited supply of Memorex Mod. 552, double density, double sided drives. These drives are rebuilt by the factory and carry a 90 day limited warranty. Delivery is stock to 15 days. Special price of $625!!!!! More than 30% off the retail price. The supply is limited so first come first served.

Mastercharge and Visa accepted on all our product line. Call and ask about our scientific systems with the Bit Pad and Plotters.

Smartbug is trademark of Smoke Signal Broadcasting, Bit Pad is trademark of Summagraphics.

See GIMIX Ad. Page 3

H H H ENTERPRISES

BOX 493, Laurel, MO.
ZIP 20810
PHONE 301-953-1155

CT-64 **CT-1024**

- ★ **FAST! Average Screen Writing Speed: 19K baud**
- ★ Memory Mapped Video Adapter for your CT-64 or CT-1024.
- ★ Just plug it in and go.
- ★ The terminal works like normal until the supplied output routine is used, then the CT-64 or CT-1024 display works at processor speed.
- ★ The J.B.I. Video Board takes 1 main SS50 slot.
- ★ Video Memory can be dip switch selected to any 1K memory slot.
- ★ Combine the J.B.I. Video Board with our up and coming pseudo graphics adapter board and you put new life into the old CT-64.
- ★ The J.B.I. Video Board comes built and is jumper selectable for either the CT-64 or CT-1024.
- ★ If your terminal is CT-1024 we need to know if it is a standard 32 characters per line; or has been modified for 64 characters per line.
- ★ **SHIPPING NOW!**

The J.B.I. Video Board sells for:
$169.00 with your 21LO2'S
$179.00 with our 21LO2'S
(8-21LO2'S are required)

We have been a dealer for SWTPC since 1976.

Johnson Micro Computer
2607 E. Charleston
Las Vegas, Nevada 89104
1-702-384-3354

Mastercharge and Visa accepted
Dealer inquiries invited

ED SMITH'S SOFTWARE WORKS

ANNOUNCES **RRMAC**

AN M6800 SYSTEM RELOCATABLE RECURSIVE MACROASSEMBLER & LINKING LOADER

ATTENTION ALL PROGRAMMERS: If you have been looking for an assembler with real macro capabilities, then RRMAC is the one for you.

Character replacement is the crux of macro expansion. RRMAC allows unlimited character replacement in any field of a macro model statement. RRMAC's argument notation can specify replacement from any field, subfield, sublist or substring of the macro call statement. RRMAC's set of macro directions supports both global and local set symbols. Set symbols can be based on either arithmetic expressions or character expressions. Macro model statements can determine attributes of set symbols or macro arguments. RRMAC allows macros to define new macros and to call other macros. Macro calls can be recursive, that is, a macro may call itself.

RRMAC contains a mini co-resident editor, allows spooling if desired, supports insert files, retains TSC Text Editor source code compatability, plus many other programmer convenience features.

RRMAC can be furnished on cassette or mini-floppy in either SSB or Mini-Flex format. Comes complete with linking loader, Instruction Manual/Programmers Guide & an extensively commented assembly listing.

M68AR . . . cassette . . .	$150.00	A 6800 to 6809 cross assembler version of RRMAC will be available in June.
M6ARR-D . . . SSB disc . . .	$150.00	
M6ARR-F . . . FLEX disc . . .	$150.00	

Specify "Mini" or Ver. 2.0

Order directly by check or MC/Visa. California residents add 6% sales tax. Customers outside of U.S. or Canada add $5 for air postage & handling.

Dealer inquiries welcome.

FLEX is trademark of TSC

Ed Smith's SOFTWARE WORKS
P.O. Box 339, Redondo Beach, CA 90277, (213) 373-3360

Software Source Books™

Combining detailed descriptions with complete source listings, these books explain the internal operations and algorithms used in Hemenway Associate's popular systems software.

How much would such a complete software resource cost? If you've seen the PAPERBYTE books by Jack Hemenway and Robert Grappel you know how inexpensive they can be. And now you can have the companion volumes to the RA6800ML macro assembler and LINK68 linking loader books.

Remember, these are not just books; they are SoftwareSourceBooks ---- complete Software resources! Order them today; VISA and MasterCharge accepted.

CP/68™ OPERATING SYSTEM ($34.95)

* PIP Peripheral Interchange
* Program transfers data between physical devices
* Wildcard Filenames and Extensions
* Relocatable anywhere in Memory
* Extended Instruction set includes 6809-type instructions (PSHX, PULX, etc)
* Device-independent I/O
* Random and Sequential Files
* Fits in less than 8K
* Chaining and overlaying
* Single Supervisor Call furnishes all DOS services
* Easily interfaced to new devices and peripherals
* Dynamic file allocation

STRUctured BAsic Language (STRUBAL+™) COMPILER for both business and scientific uses ($49.95)

* Variable precision from 4 to 14 digits
* Structured Programming forms
* Produces Relocatable and linkable code
* COMMON and DUMMY sections
* Extensibility
* String Handling
* Full scientific package
* Data structures with mixed data types

XA6809 Macro Linking Cross Assembler ($24.95)

* Runs on any M6800
* Full Macro facilities
* COMMON section for the production of ROMable code Conditional Assembly
* Generates linkable and relocatable code
* Sorted Symbol table listing
* Hash-coded Symbol table for speed

==========

Hemenway Associates Inc. 101 Tremont St. Boston MA 02108

Name	Title	Company

Street	City	State	Zip

( ) Check enclosed in the amount of $.....

( ) Bill VISA ( ) Bill MasterCharge

Card No..... Exp. Date.....

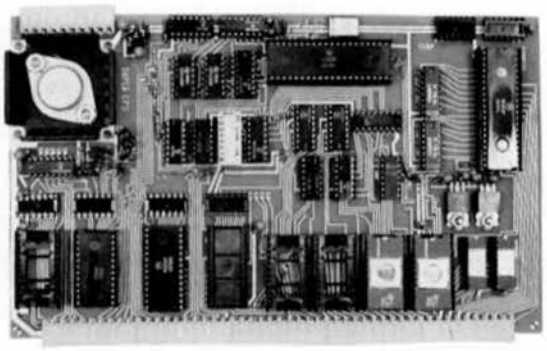
Please send the following books:

Add $0.75 per book to cover postage and handling

NOW INDUSTRIAL QUALITY AT LOW COST

FROM THOMAS INSTRUMENTATION

Industrial system boards are now available separately for OEM, prototyping or hobbyist applications. Shipped from stock, these are the same quality cards used in monitors and machine tool controls designed for GM, LTI, and General Electric. All cards are SS-50 buss compatible and are suitable for dedicated applications. The CPU card and the Video RAM Card may be combined on a TI backplane as a stand-alone micro — ideal for prototypes or hobbyists. ATTENTION OEM's: If you have a control data acquisition, monitoring, or other microprocessor application. Check with TI for more information about custom software design for the TI CPU or any other 6800 series system. TI also has non-SS-50 buss single board 6800 systems.



SS-50 SUPER CPU

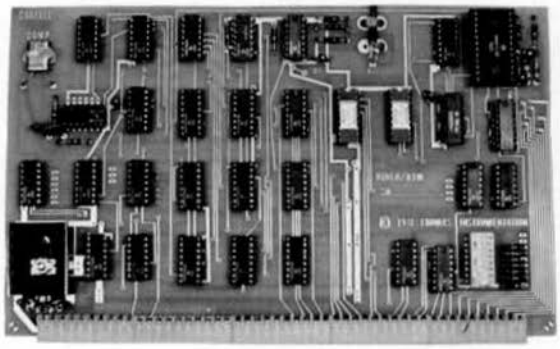
- SS-50 or stand alone computer
- 1K of RAM at $A000 I/O on board at $A400 (Relocatable)
- 2K Monitor (Mikbug compatible) in 2708 EPROM
- 2 8 bit parallel ports with 2 control bits and power
- RS-232 ACIA port. 2nd TTL ACIA optional
- 3 16 bit counter/timers (expandable to 6 add 2nd 6840)
- 128 byte RAM at 0000 is jumper selectable
- Battery back-up for 32 bytes of RAM
- Plug back to back with Video RAM for Stand alone micro or customized smart terminal

ASSEMBLED $195.00
CARD AND DOCUMENTATION $49.00

SS-50 VIDEO RAM

- Fully synchronous operation — No jitter
- 7 by 9 Characters Programmable reverse video
- Full 128 Character ASCII set
- 1K of memory can be mapped to any 1K boundary
- Full documentation includes software (Replaces OUTEE)

ASSEMBLED $149.00
CARD, CRYSTAL and DOCUMENTATION $39.00



TI SS-50 Wire-Wrap Card	**24.00**
TI SS-50 Parallel I/O Card	**95.00**
Card only	**35.00**
GIMIX 16K Static RAM w/Soft addressing	**368.00**
GIMIX 16 RAM without Soft addressing	**298.00**
3, 4, 7 SLOT Backplanes (per slot)	**4.00**

TI cards available from stock

THOMAS INSTRUMENTATION

168-8th Street, Avalon, N.J. 08202
Phone (609) 967-4280

VISA

DEALERS FOR GIMIX SWTPC SSB

CALL FOR DEALER, OEM. AND QUANTITY PRICES

SMOKE SIGNAL BROADCASTING PRESENTS...

THE SMOKE WRITER (VDB-1)

The SSB SMOKE WRITER incorporates the latest advances in electronic technology to bring you and the SS-50 bus a truly unique video display board. The SMOKE WRITER uses the MC6845 CRT Controller chip and provides total control over the display format.

IT'S NEW

$349.00

The standard features of the SMOKE WRITER are:

- *80 x 24 display with 32 graphic characters. Optional character generator ROM with 128 ASCII characters plus 128 graphic characters.*
- *Upper and lower case characters with lower case descenders.*
- *Programmable character set, a total of 128 characters in a 2K EPROM; a 256 character 4K EPROM is optional.*
- *1K EPROM for Software drivers.*
- *Reduced intensity or reversed video*
- *Programmable display rate (10 to 5000 character per second) equivalent to 100 to 50K baud.*
- *Protected fields.*
- *Addressable Cursor.*
- *2K video display RAM accessible by the CPU as standard RAM Memory.*
- *128 Bytes of Scratch pad RAM.*

If you have a need for a fast and dependable video display board, Smoke Signal Broadcasting has what you are looking for. The SMOKE WRITER is right at home when used with a cursor based editor or in a business program that needs protected fields.

SMOKE SIGNAL  **BROADCASTING**

31336 Via Colinas, Westlake Village, CA 91361, (213) 889-9340

'68' Micro Journal
3018 Hamill Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025